VOLUME 05 | ISSUE 04

digit FastTrack

YOUR HANDY GUIDE TO EVERYDAY TECHNOLOGY

Creating Facebook apps

to

Get famous, and earn money by creating a killer Facebook App. We provide the pills of wisdom you need.

CHAPTERS

Fast Track to Creating Facebook Apps

digit

YOUR TECHNOLOGY NAVIGATOR

thinkdigit.com

Fast Track

to

Creating Facebook Apps

facebook

thinkdigit.com

Credits

The People Behind This Book

EDITORIAL

Editor	Robert Sovereign-Smith
Head-Copy Desk	Nash David
Writers	Rahil Banthia

DESIGN AND LAYOUT

Lead Designer	Vijay Padaya
Senior Designer	Baiju NV
Cover Design	Jayan Narayanan

Contents

Contents

Introduction

Facebook has grown phenomenally over the past several years from an Ivy League social web application to one of the top ten most visited web sites on the internet. It currently has more than 250 million active users and more than 350,000 active applications with billions of page views per month.

It comes with its own platform for application development which consists of an HTML-based markup language called Facebook Markup Language (FBML), an application programming interface (API) for making representational state transfer (REST) calls to Facebook, a SQL-styled query language for interacting with Facebook called Facebook Query Language (FQL), a scripting language called Facebook JavaScript for enriching the user experience, and a set of client programming libraries. Facebook applications are hosted on the application developer's own server, but because of the way they interface with Facebook, they appear to users to be part of Facebook itself. Using Facebook applications, developers can add custom features and new ways for users to interact with each other using Facebook.

We'll help you get started on building powerful Facebook applications, and details how to use the core technologies of the Facebook Platform to create an application, how to deploy, monitor, and tune it, how to work with data stored in Facebook, including photos, how to handle multimedia and other custom data within Facebook applications, how to send notifications and invitations from within an application, update a user's profile, how to create application control panels for users, and much more.

Hopefully, this book will go a long way into making you understand what makes applications work on Facebook, how to monitor their progress, and make changes that maximise their potential for success. This book will not only act as a reference for the various parts of the Facebook Platform, but also address the entire process of creating applications from inception to deployment and beyond. Besides taking you through the necessary steps to getting your application built in the best way possible, this book also teaches you how to maximise your chances of getting your application to spread successfully and make money out of it.

Chapter 1, 2, and 3 introduce you to the basic Facebook application architecture, integration points, and the elements used by your application to interface with Facebook, i.e., the Facebook Platform that consists of the Facebook API, FBML and FQL. It acts as a primer to making successful Facebook applications and a quick reference to the key points in Facebook application development.

Chapter 4, 5 and 6 explains the tools of the Facebook platform in detail, and analyse the technique used to work with them, using various examples. This will cover the essentials needed to get you started on application development, along with setting up a developer environment, getting the necessary tools and how to use and test your code using the test consoles on Facebook. You will also be shown the scripting ability in Facebook, using a subset of standard JavaScript functions in FJS. It will also serve as an introduction to Facebook Query Language (FQL), which is sometimes more effective to use than the Facebook REST API

Chapter 7 will cover Profile pages and how to extract the Facebook information to be displayed by your application, store information in the Facebook cache and display it on your users' Profile pages, and store your own custom data.

Chapter 8 will cover the details of managing feeds (both news feed and mini feed) using Facebook REST APIs. It will also cover Publisher, which is very similar to Feeds.

Chapter 9 will focus on creating a successful invitation and notification system, with source code snippets on how to create a successful notification system, use notifications efficiently, and send emails.

Chapter 10 will detail the methods for spreading your application, measuring its success and making money out of it. It will also provide you help resources and references in case you bump into any problem during development. d

1 The Facebook platform

The Facebook platform consists of various elements that allow application developers to build an application and replicate the look and feel of Facebook within it. These elements are mostly derived from well-known web development languages such as HTML, SQL and JavaScript for easy portability. They are:

The API (Application Programming Interface), also known as the REST API that handles communication and function calls between Facebook and your application.

- FBML (Facebook Markup Language), a markup language derived from HTML
- FQL (Facebook Query Language), for data handling.
- Facebook JavaScript, for scripting needs.
- Client libraries for programming languages.

Let us look at all of these components in detail to understand the need and use of them while making powerful yet simple Facebook applications.

1.1 FBML (Facebook Markup Language)

It is a tag-based programming language like HTML used to format your information, albeit a bit more fancy as each interaction starts and ends with a tag. FBML provides a large set of Facebook user interface and programmatic primitives required to abstract complex code and make most routine procedures effortless. The tags are automatically parsed and translated into HTML, CSS (Cascading Style Sheets), and JavaScript code by Facebook servers when a request for an application page that contains it is detected. If you've previously used HTML, then you will have no problems adapting to FBML. To distinguish between HTML and FBML commands in your application code, you need to prefix FBML tags with fb: as you would if you were using multiple DTDs/schemas in XHTML.

For example, if you want to add a link to your application's about page on your dashboard, all you need is to add the following lines of code in your application:

```
  <fb:dashboard>
       <fb:about href="about.php">About the Application</
fb:about>
  </fb:dashboard>
```

1.2 FQL (Facebook Query Language)

FQL stands for Facebook Query Language. It offers a lot of SQL-like features and language elements that allow applications to directly query Facebook's internal data tables. FQL uses the same syntax as typical ANSI-SQL, so if you've worked with SQL before, FQL isn't a big deal. FQL is powerful in the sense that it accesses and returns the same data provided by many of the Facebook API calls; however, it allows applications to have Facebook filter that data with a language of your choice before it's returned to the client, which potentially speeds up page loading and response times.

For example, if you want the name, and birthday of all of your friends, then you can use the following query in the API Test Console:

```
SELECT first_name, last_name, birthday
FROM user
WHERE uid IN (   SELECT uid1
                 FROM friend
                 WHERE uid2=1234646)
```

Don't forget to replace our ID (1234646) with your own because you can only use the details of friends of the logged in user. We are still using the API, but only one method—fql.query. This basically means that you're not limited to the information supplied by the API methods and the FBML tags. You can actually extract the exact data that you want with whatever language you are using.

1.3 FBJS (Facebook JasaScript)

FBJS is the Facebook's version of JavaScript for developers who really want, or need, to use JavaScript in their applications. Facebook scrubs (removes) much of the JavaScript you add to your application to minimise the threat of cross-site scripting (XSS) attacks, but by using Facebook JavaScript (FBJS) you can still enrich the user's experience. It supports most of the DOM-based manipulation methods of JavaScript and the familiar events, functions, anonymous closures, and properties. If you're familiar JavaScript, you'll pick this up quickly (or perhaps find it maddening!). For example, if you want to provide a modal dialog box to your users:

```
  <a href="#" onclick="new Dialog().showMessage('Dialog',
'Message
  for this link');return false">Show Dialog Box</a>
```

When this code gets processed by the Facebook platform, a user will be shown the modal dialog box after clicking the Show Dialog Box hyperlink. Pretty good for a single line of code! It however differs from JavaScript in many ways. The syntax is slightly different, primarily to protect Facebook itself from malicious JavaScript code and cross-site scripting (XSS) attacks. Many DOM properties used in normal JavaScript are replaced by get/set property methods. Finally, some event handlers are not available that are widely used in normal client-side JavaScript. The use of the ubiquitous `onload()` event handler to execute code when a web page loads is not allowed. A user must take an action on the page (set focus to a control, click a mouse button, hit a key) before FBJS can execute in most locations. Facebook also wraps all FBJS in a sandbox. Basically, all FBJS variables, function names, and function parameters are prepended with a special string that ensures that no FBJS code conflicts with or can override any existing JavaScript code that might be on the page.

1.4 XFBML and the Facebook JavaScript Client Library

Applications which are based on IFrames and external sites that use Facebook Connect do not have access to FBML because of platform limitations and, therefore, don't get access to a lot of the Facebook controls and widgets it provides. To overcome these limitations, Facebook provides XFBML as well as the Facebook JavaScript Client Library which give developers access to most of the features provided by FBML. Applications that use XFBML render their pages using strict XHTML and load the Facebook JavaScript Client Library to get access to its features. These applications use the Facebook API on the client rather than the server. This means that a Facebook application can run anywhere and not just inside Facebook. Features such as logging in with Facebook credentials and publishing news stories are available in any web browser thanks to this.

After the page loads and initializes the Facebook JavaScript Client

Library, it can call the API functions and use XFBML which can either be displayed inline on the page or dynamically created via the library.

1.5 Application architecture

All Facebook applications have the same basic architecture to interoperate with the Platform. For this reason, Facebook provides a Facebook Developer application to allow developers to make new applications and add in the details where they are hosted. Each application is issued a secret identifying keys, and developers are required to enter URLs for how users and Facebook access the application.

This does not mean that Facebook actually host developer's applications. It simply acts as a proxy instead so that when the user visits the Canvas Page URL, Facebook creates an outer frame and then calls the application's Canvas Callback URL to get information to display.

1.5.1 Secret Keys

Whenever an application is created on the Facebook Developer platform, it is given a public and private key pair. The public key is called the API key, and the private key is called the Secret key. These keys verify that all calls made to the Facebook API are from that very application. It is an important security feature and developers should protect their Secret keys and report their loss if they are ever compromised. Otherwise, any other application can start making calls masquerading as that application, and modify or even delete user data.

1.5.2 Canvas Page URL

The Canvas Page URL is the URL that Facebook users use to get to the application. It is of the format http://apps.facebook.com/canvaspageurl. It needs to be unique.

1.5.3 Canvas Callback URL

The Canvas Callback URL is the URL of your server where your application resides. Facebook makes calls to this URL whenever it needs to display an application page, or when a user adds or removes the application, or when it needs to update its cache. The server hosting the Canvas Callback URL should be able to handle the callbacks using whatever web server or language the developer deems appropriate. Facebook sends the set of data as POST variables to the Canvas Callback URL, which contains information about the viewing user, the session, and the application.

1.5.4 Canvas Page Workflow

Whenever a user accesses an application canvas page, Facebook calls the application's Canvas Callback URL. Your Facebook applications can either have FBML or IFrame based canvas pages. The biggest difference here is that applications that produce FBML requires Facebook to render their content before displaying it. This is so that Facebook can convert each FBML control into its HTML and JavaScript equivalents for your web browser. IFrame-based canvas pages just show their content directly.

1.6 Facebook Mobile Support

Facebook provides rich mobile integration for most mobile phones and even supports the iPhone with the Facebook Connect for iPhone library. The integration points are generally the Facebook's mobile web site – http://m.facebook.com or the use of Facebook's Short Message Service (SMS, or text messaging) service.

Mobile applications for the mobile Facebook web site use only a subset of FBML to create mobile profile and canvas pages. A special fb_sig_mobile parameter is passed to the application as a POST variable whenever a mobile browser is detected by the Facebook servers. The application then returns only the FBML enclosed in special `<fb:mobile>` tags for display. Mock AJAX and FBJS are yet not available in this version.

The user needs to go to the Settings page on http://m.facebook.com and enable the application to show up on her mobile profile even after he has added the application.

Facebook's SMS service lets applications send and receive SMS notifications. Again, the user must first allow this feature by enabling the SMS extended permission in the application. To check whether a user has set permissions, you can call the `sms.canSend()` API function. Once the permissions have been granted, applications can send an SMS notification or start an SMS conversation, depending on the parameters to the API call. The user can also reply to the SMS or follow a link inside of it to the application's mobile Facebook Page.

The API wrapper code for handling SMS calls is not yet part of the PHP client library. However, Facebook Connect for the iPhone library allows iPhone developers to include an Xcode project into their applications. iPhone applications will be able to access the Facebook Platform API to get user information, set a status, create a news story, and get friend information with the help of this. d

Win!!

Exciting Prizes worth Rs.20000

What you learn from this Fast Track could win you Rs. 20,000 in exciting prizes. Take part in Digit's Facebook Apps contest by going to

www.thinkdigit.com/d/fb_contest

2 The Facebook API

The Facebook API (or Application Programming Interface) is a set of software libraries that enable you to work with Facebook without knowing anything about its internal workings. All you have to do is obtain the client libraries, and start making use of them in your own application.

2.1 The classification

Facebook API calls are grouped into various action categories, each of which focuses on a different aspect of the Platform. These methods are really wrappers for more sophisticated FQL interactions with the Facebook back end but are useful bits of code that speed up the development of your application. Before we get into the details of these categories, let us look at some basic ones:

- facebook.auth - basic authentication checks for Facebook users.
- facebook.feed - post to Facebook news feeds.
- facebook.friends - query Facebook for various checks on a user's friends.
- facebook.notifications - send messages to users.
- facebook.profile - set FBML in a user's profile.
- facebook.users -information about your users (such as content from the user's profile and whether they are logged in).
- facebook.events -ways to access Facebook events.
- facebook.groups - access information for Facebook groups.
- facebook.photos - interact with Facebook photos.

Facebook consistently modifies these APIs as part of the REST API as new features are added, security issues are addressed, or behaviour deprecates or becomes obsolete.

Authentication API

The Facebook REST API has two methods facebook.auth.createToken and facebook.auth.getSession for dealing with authenticating your users. `facebook.auth.createToken` is the method that creates an authentication token (`auth_token`) that is then passed to the Facebook authentication mechanism. Once the user is logged in, the second method, facebook.auth.getSession will contain this token in the response if you specifically request the auth_token in the response.

Because Facebook takes responsibility for these actions, you don't have to take the headache of authentication and purchase SSL certifications, implement your own encryption schema for passwords, or even worry about sessions. In our case while working with the PHP client library, you start the authentication procedure by calling the Facebook object's require_login method. Your users are then redirected to Facebook's login page (https://login.facebook.com/login.php), which is passed with your API key, and the user is given a session key and redirected to your callback page. When the user enters the application for the first time, he is asked to accept the terms of service for your application.

Instead of logging into Facebook every time you want to update the data to use some sort of scheduled task, you can do so with an infinite session key. The process to get your infinite key is however, a bit more convoluted. After you've made your application, create a new page (`infinite_key.php`) on your server, i.e., your callback domain. This will create a new Facebook object and echo your session_key:

```
<?php
$facebook_config['debug'] = false;
$facebook_config['api_key'] = '<your_api_key>';
$facebook_config['secret_key'] = '<your_secret_key>';
require_once('<path_to_api>/facebook.php');
$facebook = new Facebook($facebook_config['api_key'],
$facebook_config['secret']);
$user = $facebook->require_login();
$infinate_key = $facebook->api_client->session_key;
echo($infinate_key);
?>
```

Now log out of Facebook, clear your Facebook cookies and browser cache, and then go to the page you just created on your server. Once you've logged on, you should see the infinite key that you can then use in your code. You can now use your own UID and the key just obtained in other code to perform anything that needs to happen on a regular basis with the set_user function in the facebook object:

```
<?php
...
$uid = '<your_uid>';
$key = '<your infinite key>';
$facebook->set_user($uid, $key);
// other code
?>
```

Permissions API

The permissions API contains functions and methods to manage applications' developer settings, retrieve application Facebook metrics, ban specific users, automate application configuration and setup on different servers, and get application public information to control overall application management.

Authorization API

The Authorization API contains functions and methods used for handling session management and login information on both desktop or external Facebook applications.

Batching API

The slowest part of a web application is waiting for the data requested from some remote source to be returned because communicating with a remote server via HTTP or any protocol is expensive in terms of response time and latency. Facebook applications fall in the same category and are no exception to this. The Batching API allows applications to bundle up to 20 API calls and make a single call to a remote Facebook server instead of several individual ones. This significantly speeds up that part of a Facebook application. It is even configurable in the sense that these calls can be made sequentially or in parallel, depending on the application's needs.

Comments API (Beta)

The Comments API includes methods that allow applications to comment on individual Feed stories and add it to the stream along with the ability to get and remove these comments programmatically.

Data Store API

The DataStore API provides methods for basic database manipulations offline caching of data with basic create, read, update, and delete (CRUD) calls for storing data that you access through REST. This follows the object-oriented database management systems (OODMSs), which is a bit different from the relational database management systems (RDBMSs) in terms of terminology.

To use the Data Store API, you need to define your schema (your database), which consists of object types (tables) and properties (columns). Facebook's servers perform your database manipulations for you. However, there aren't any structured queries, full-text search, or transaction-level query processing in the Facebook Data Store API.

There are three basic functions in the DataStore API: specialized tables, distributed tables, and associations that are split into five separate APIs (User Preference, Object Data Definition, Object Data Access, Association Data Definition, and Association Data Access).

According to Facebook, it can handle millions of records with little or no performance degradation, something we can't be sure of about data stored on local servers. Because indexing tables in a distributed environment won't necessarily provide a performance boost, Facebook provides a specialized table of users that is optimized to provide performance for fast lookups (such as indexes) with the help of the associations component of this API. This mechanism has been implemented to provide fast lookups without centralized indexes or parallel queries. The user preferences for the API are a list of 128-character strings, for which you can store up to 201 for each user (numbered 0–200). Access to the getters/setters methods are accessed through getters and setters in the REST API (`facebook.data.setUserPreference` and `facebook.data.getUserPreferences`).

Data objects can be created with `facebook.createObjectType`. These objects takes a name and contains a set of object properties like data types. However, you don't have the same type of control over the data types with this API as you do with normal RDBMS because you are limited to integers, strings (less than 256 characters), and text blobs (with a maximum of 64 KB).

You create, read, update, and delete through the Object Data Access API after you have defined your objects and object types. These go like `facebook.data.createObject`. But, you first need to define the relationship between objects in the `facebook.defineAssociation` call to work with the associations between objects. There are two basic types of associations: one-way, symmetric associations and asymmetric associations. The asymmetric association is like a many-to-many join and a symmetric association is like a one-to-many join. You can then create the actual associations with the help of the Association Data Access API which contains methods required to create, remove, and retrieve these associations and retrieve the object details from the data contained in the data definition.

Example:

```
  <?php
  $createObject    =    $facebook->api_client->data_
createObjectType("wallpost");
  $uid = $facebook->api_client->data_defineObjectProperty
("wallpost", "uid", 1);
```

```
  $time = $facebook->api_client->data_defineObjectPropert
y("wallpost", "timePosted", 2);
  $post = $facebook->api_client->data_defineObjectPropert
y("wallpost", "post", 3);
  ?>
```

This code snipped is analogous to the SQL:

```
CREATE TABLE wallpost(
uid integer,
timePosted timestamp,
post text
)
```

It takes a little bit to get used to the DataStore API because you're not performing typical SQL DDL; but once you get yourself around creating the objects, it becomes relatively trivial to use the API as the persistence layer for your application. This API is a boost for those who choose to have Facebook manage their data instead of writing their own SQL queries and using the client library to manage data.

Events API

Facebook has rich event system, which allows users to create events with built-in RSVP features, event-specific media, and export capabilities. The events API allows applications to manage events on its users' behalf with two main wrappers to retrieve information on events. `facebook.events.get` method returns a response based on the filters passed to it and the `facebook.events.getMembers` method returns the RSVP status for Facebook members for a given event. These methods requires extra permissions from the user.

FBML API

The FBML API is a set methods that allow developers to create custom FBML tags, refresh cached images, upload localized strings for internationalisation, and do high-performance updates of cached profile box FBML. The `facebook.fbml.refreshImgSrc` method is used to fetch and catch an image at a given URL. The `facebook.fbml.refreshRefUrl` method is used to refresh content from a given URL and the `facebook.fbml.setRefHandles` method is used to get content as a handle in FBML. You can even register custom FBML tags with Facebook and make them public so that other developers can use them in their own applications.

Feed API

The Feed API contains methods that allows applications to register templates for the stories they publish to the stream and to directly publish these stories programmatically. To publish information to an individual user's feed, use the `facebook.feed.publishStoryToUser` method and the information will get published to a user's feed based on their session key (`session_key`) and their user ID (`uid`). The `facebook.feed.publishActionOfUser` method can be used to publishe a mini-feed story to the user's account, also based on their session key and user ID.

Friends API

Friends API allows your application to look at the friends of your users. The method `facebook.friends.areFriends` can be used to get whether two people are friends. The `facebook.friends.get` method can be used to return a structure of the user's friends' user IDs. Also, the `facebook.friends.getAppUsers` method can be used to return a structure of the friends of the user who have installed the application on which you're working.

FQL API

FQL API allows calls which are wrappers for complex, but common, FQL queries so that developers can access information provided by API methods, but by using a SQL-like query language. Many of Facebook's API methods use FQL internally to access the data they return or need. In general, it is more efficient to use a FQL query than an equivalent API call. The facebook.fql method will take a query and return a response object based on your syntax.

Groups API

If you want your application to work with your user's groups, the Groups API has two methods that return information about their visible groups. You can either retrieve all the groups the user belongs to or a subset of these groups with the `facebook.groups.get` method or use the `facebook.groups.getMembers` method to return the user IDs (UIDs) for a given public group.

Links API

The links API contains methods that allow applications to give its users the ability to post links to their Walls directly from the application. These API method calls work in a similar way as the ubiquitous Facebook Share button.

User Management API

The User Management API is probably the most frequently used set of APIs in the Facebook Platform, because it provides your applications direct access to the users and their information in the social graph. These APIs contain methods required for getting users' friends, individual user's information, verifying if specific users have authorised your application, and organising lists of friends. `facebook.users.isAppAdded` method tells you whether the user has added your application. If you with to get information from your user's profile, you can call the method `facebook.users.getInfo`. To get the user ID (`uid`) from a session key, you need to use the `facebook.users.getLoggedInUser` method.

Marketplace API

Facebook has its own marketplace where users can buy/sell items, list jobs, list housing, or even request items to purchase/borrow. To enable your application to search the Facebook marketplace, use the `facebook.marketplace.search` method. To get/set listings, the `facebook.marketplace.getListings` and `facebook.marketplace.createListing` are helpful. You can even remove your listings with `facebook.marketplace.removeListing`. Furthermore, if you want your application to be able to search within certain specific categories, the API provides you `facebook.marketplace.getCategories` and `facebook.marketplace.getSubCategories` methods.

Notes API

Facebook lets your application create, edit and delete rich text documents called Notes if it has been granted special permissions. We will cover this is more detail in a subsequent section.

Notifications API

With its REST API calls, Facebook allows you to send and receive notifications in your application. To get your user's request notifications and to see outstanding notification responses, use the facebook.notifications.get method. To send notifications to your users, use the facebook.notifications.send method and to send invitations to them, use the facebook.notifications.sendRequest method.

Messaging API

Messaging API contains the API methods that allow applications to send Facebook notifications and email (with the help of the Notification API), update users' Facebook status (with the help of the Status API), and send Live Messages, i.e., messages sent to a particular user's browser via FBJS that are extremely useful in applications that have sequential or turn-based semantics, such as games.

Pages API

Pages API contains methods that allow developers to get specific information about Facebook Public Profiles or previously known as Facebook Pages. They return information about whether a logged-in user is the administrator of the Public Profile, if he/she is a Fan, and whether a specific application has been added to the profile.

Photo and Video API

Facebook is one of the largest photo and video sharing web site on the web. More than 60 million images are added every week by its users. To handle these, Facebook provides a set of APIs for developers to manage and interact with these assets from within their applications. These methods and functions allow users to upload, create albums, and retrieve tagged user information, among other things. Images can be tagged with the `facebook.photos.addTag` method. Photo albums can be created by using the `facebook.photos.createAlbum` method. To get a user's individual photos, use the `facebook.photos.get` method and for a listing of their albums, use the `facebook.photos.getAlbums` method. The listing of the tags that individual photos have can also be obtained with the `facebook.photo.getTags` method. You can also upload photos with the method `facebook.photos.upload`. Many of these methods require extended permissions for applications to use them.

Profile API

The profile API contains methods to easily interact with setting information in the user's profile and manage users' application profile boxes, application tabs, or info sections. The two methods you will be using as part of this are `facebook.profile.setFBML` and `facebook.profile.getFBML`.

Open Stream API

With Facebook's real-time streaming update model, publishing content to the stream is currently a somewhat complex and detailed process. The new Open Stream API provides a new way for applications to access the

Feed than what's provided normally. It even lets you access the stream, read content from the stream, publish your own content, and manage comments and ratings for individual Feed stories from outside Facebook, whether it be on an external web site, a mobile application, or a desktop application.

2.2 Test console

You can learn a lot about the API's functionality by using the Facebook API Test Console. Go to the URL http://developers.facebook.com to access the Facebook developers' section in your browser. Click on the link to the Tools page or go directly to http://developers.facebook.com/tools.php.

The API Test Console has a number of felds:

User ID - A read-only field which displays your user ID number.
Response Format - This is used to select the type of response that you want:

- XML
- JSON
- Facebook PHP Client
- Callback - With this, you can encapsulate the response in a function if you are using XML or JSON.
- Method - To insert the actual Facebook method that you want to test.
- The Test Console also has the documentation link which is used to select a method from the drop-down list, and view its documentation which consists of:
- A description of the method
- The parameters used by the method
- An example return XML
- A description of the expected response.
- The FQL equivalent
- Error codes

API Essentials

API calls are your first interaction points with the platform. You don't need to use the client library; but it just makes things easier since you would need to write these in your language in order to interact with the platform.

To set up the client library for PHP, just include the client library in your code.

For example, the code to make sure that users are logged on to Facebook before displaying a list of the user's friends goes like:

```
  <?php
  $facebook_config['debug'] = false;
  $facebook_config['api_key'] = '<your_api_key>';
  $facebook_config['secret_key'] = '<your_secret_key>';
  require_once('<path_to_client_library>/facebook.php');
  $facebook = new Facebook($facebook_config['api_key'],
  $facebook_config['secret']);
  $user = $facebook->require_login();
  $friends = $facebook->api_client->friends_get();
  echo  "<pre>Friends:"  .  print_r($friends,  true).  "</
pre>";
  ?>
```

Here, you're just throwing the results from the friends object on the screen. The syntax for interacting with the API is abstracted into methods of the API in the Facebook object. This code requires the user to be logged on to Facebook in order to execute. The `require_login()` method is used to store the user's identification number (UID) in the variable user. The next call wrapped it in the PHP method friends_get queries the Facebook API method friends.get. There's no need to pass the UID to the API request because the facebook object holds the user's identification number (their primary key or UID). The PHP code here actually sets up a REST client to call the `facebook.friends.get` API method and returns an array of user identifiers:

```
  public function friends_get() {
  if (isset($this->friends_list)) {
  return $this->friends_list;
  }
  return    $this->call_method('facebook.friends.get',
array());
```

3 Official libraries

Facebook provides official support for the following libraries:

- **PHP 5** – use the subversion-command `svn checkout http://svn.facebook.com/svnroot/platform/clients/php/trunk/`. It is the most widely used library by developers. Most books and Developer forum posts reference the use of this library.
- **JavaScript Client Library** – Although it's not as complete as the PHP library, it forms the basis of Facebook Connect and will be expanded as time goes on.
- **Facebook Connect for iPhone** – http://www.github.com/facebook/facebook-iphone-sdk. It is written in Objective C, which is the iPhone's primary development language. This library provides services and methods to make it easier for iPhone developers to allow their applications to connect to Facebook from the iPhone.
- **ActionScript 3.0 Library for Facebook Platform** – http://www.adobe.com/go/facebooklibrary. It is officially supported by Adobe. This library allows Adobe Flash, Flex, and Air applications to directly refer and call the Facebook API.
- **Microsoft SDK for Facebook Platform** – http://www.microsoft.com/facebooksdk. This is officially supported by Microsoft.
- **Force.com for Facebook** – It is officially supported by salesforce.com. This library allows developers on the Salesforce platform to integrate Facebook into their custom Sales force platform applications.

Unofficial Libraries

The Facebook Platform developer community maintains the following client libraries. Facebook does not provide official support for them. Many of them are active, but some are just a proof of concept.

- Android
- FBFlashBridge
- ASP.NET
- ASP.NET MVC
- ASP (VBScript)
- ASP (JScript)
- Cocoa
- ColdFusion - See also Developing a ColdFusion Application in 8 Minutes
- C++

- C#
- D
- Emacs Lisp
- Erlang
- Google Web Toolkit
- Haskell
- Java
- Kontagent pre-instrumented libraries - Provides Instant to Kontagent Social Analytics without additional work
- PHP - with Kontagent Analytics - Pre-Instrumented with Kontagent – a fbFund Company
- Ruby on Rails - with Kontagent Analytics – Pre-Instrumented with Kontagent – a fbFund Company
- Facebook Connect for Web with Kontagent Analytics – Pre-Instrumented with Kontagent – In Beta
- Facebook Connect for iPhone with Kontagent Analytics – Pre-Instrumented with Kontagent – In Beta
- Lisp (Common Lisp) – seems to run only with SBCL
- Perl
- Python
- Ruby
- Ruby MiniFB gem – a tiny, simple to use Ruby library for Facebook
- Ruby on Rails
- Smalltalk
- Tcl
- VB.NET
- Windows Mobile/Facebook Development with the .NET Compact Framework
- zembly, allows executions within zembly and external invocations from Java, JavaFx, and PHP

Facebook updates its libraries frequently, but these changes are generally always preannounced on the developer blog. In these changes, API methods are often deprecated but they still function correctly until a cutoff date. This means that older applications do not necessarily have to be updated or rewritten to always use the latest version of the library.

Client Library Essentials

We shall be using the PHP library throughout the Facebook. If you're planning on working with some other language, most of the concepts will be the same, but depending on how the developer who wrote the library chose to implement the elements and the characteristics of the Facebook API, the way methods are called and named might be slightly different. Also, some libraries are better documented than others; but in case you run into a problem and you can't get help with your language-specific version, take a look at the documentation for the "official" client libraries for PHP or Java.

Facebook's PHP library consists of two main object classes, Facebook (`facebook.php`) and Facebook Rest Client (`facebookapi_php5_restlib.php`). The Facebook Rest Client class abstracts interactions with Facebook's API. The Facebook class just utilises the FacebookRestClient class's methods to further abstract common interactions with the Facebook platform. For instance, instead of writing a function to require a user to have a session key to work with your application, the Facebook object has a method called `require_login()` that checks for the session key and, if it doesn't find one, redirects the user to Facebook to perform the authentication on the login page and then returns the user to your application.

Example usage:

```
<?php
require_once('<client library>/facebook.php');
$secret_key = '<secret key>';
$api_key = '<apikey>';
$facebook = new Facebook($api_key, $secret_key);
?>
```

The API and secret keys are assigned to you in your My Applications portion of Facebook's Developer application (http://www.facebook.com/developers/apps.php). Once instantiated, the Facebook client libraries make it easy for your application to interact with the Facebook platform.

When you create a Facebook object, that class includes a library to make the REST calls (`facebookapi_php5_restlib.php`). If you're using PHP to write a desktop application, you need to use the `facebook_desktop.php` file, which extends the Facebook object but is better suited to desktop applications. The `facebookapi_php5_restlib.php` file is where you will find most of the functions you will use in your application and thus is the real workhorse for your application.

Let's take an example to see how all the different parts and components of the program work together with the platform. This sample application allows users to set their status using the PHP client library, the Facebook

API, FQL, mock Ajax, and FBML. Create an index.php file in the Facebook folder of your web server and use the following source code:

```
<?php
$facebook_config['debug'] = false;
$facebook_config['secret_key'] = '<secret key>';
$facebook_config['api_key'] = '<api key>';
require_once('<client library>/facebook.php');
$facebook = new Facebook($facebook_config['api_key'],
$facebook_config['secret']);
$user = $facebook->require_login();
?>
```

This sets up the constants (api_key and secret_key) and instantiates a Facebook object (which includes an instance of the API REST client). To add functionality to the application, use the following code block:

```
<div style="padding:5px;">
<h1>Hello
<fb:name   uid="<?=   $user?>"   firstnameonly="true"
capitalize="true" />
</h1>
<p>What's your new status?</p>
<form>
<input type="text" name="status" value="confused" />
<input type="submit"
clickrewriteid="curr_status"
clickrewriteurl="<callback url>" />
</form>
<div id="curr_status"></div>
</div>
```

The user variable you set in the PHP code is used to display the user name. We've also created a mock Ajax call to populate the empty preview div. Code to handle updating the status goes like:

```
if(isset($_REQUEST['status']))
{
$has_permission =
$facebook->api_client->call_method("facebook.users.
hasAppPermission",
array("ext_perm"=>"status_update"));
if($has_permission){
  $facebook->api_client->call_method("facebook.users.
setStatus",
  array("status" => $_REQUEST['status']));
  $fql = "SELECT status FROM user WHERE uid=" . $user;
  $query_result = $facebook->api_client->fql_query($fql);
  $status = $query_result[0]['status']['message'];
  echo("<p>Your new status is: <strong>" . $status . "</
strong></p>");
  } else {
  $url = '<a href="http://www.facebook.com/authorize.
php?api_key=';
  $url += $api_key . '&v=1.0&ext_perm=status_update">Click
here</a>';
  echo('<p style="font-size:14px;"> ');
  echo('D\'It doesn\'t look like you have the permissions
to change yourstatus. ' . $url . ' to add the permissions
to update yourstatus.');
  echo('</p>');
  }
  exit;
  }
```

What we are basically doing is first checking the permissions for the user because updating the user's status requires additional permissions and if the user has permission, updating the status using an APIcall, and then you querying Facebook with FQL to make sure that the status you just set is in fact the current status. Next we are displaying it within the curr_status div. The source code becomes:

```
  <?php
  $facebook_config['debug'] = false;
  $facebook_config['secret_key'] = '<secret key>';
  $facebook_config['api_key'] = '<api key>';
  require_once('<client library>/facebook.php');
  $facebook = new Facebook($facebook_config['api_key'],
  $facebook_config['secret']);
  $user = $facebook->require_login();
  if(isset($_REQUEST['status'])){
  $has_permission = $facebook->api_client->call_method(
  "facebook.users.hasAppPermission",
  array("ext_perm"=>"status_update")
  );
  if($has_permission){
  $facebook->api_client->call_method(
  "facebook.users.setStatus",   array("status"   =>   $_
REQUEST['status'])
  );
  $fql = "SELECT status FROM user WHERE uid=" . $user;
  $query_result = $facebook->api_client->fql_query($fql);
  $status = $query_result[0]['status']['message'];
  echo("<p>Your new status now: <strong>" . $status . "</
strong></p>");
  }else {
  $url   =   '<a   href="http://www.facebook.com/authorize.
php?api_key=';
  $url += $api_key . '&v=1.0&ext_perm=status_update">Click
here</a>';
  echo('<p style="font-size:14px;"> ');
  echo('D\'It doesn\'t look like you have the permissions
to change yourstatus. ' . $url . ' to add the permissions
to update yourstatus.');
  echo('</p>');
  }
  exit;
  }
  ?>
  <div style="padding:5px;">
  <h1>
  Hello
  <fb:name   uid="<?=   $user?>"   firstnameonly="true"
capitalize="true" />
  </h1>
  <p>What's your new status?</p>
  <form>
  <input type="text" name="status" value="confused" />
  <input type="submit" clickrewriteid="curr_status"
  clickrewriteurl="<callbackurl>" />
  </form>
```

```
<div id="curr_status"></div>
</div>
```

The Facebook platform has lots of quirks since the code you write is interpreted through Facebook. Whenever you create a new form, Facebook creates several additional input fields in your form, which are used by Facebook to process your input. Another thing to keep in mind is that FBML isn't HTML. Although you can use most of the familiar tags in FBML as you do in HTML, there are differences in the way in which tags are processed. Your ability to use external CSS files for your application is also limited. You can only embed your CSS in the page using the `<style>` tag. Generally it isn't a big deal if you have a small application, but as your application grows in size and usage, this becomes less manageable. An alternative to this is to "fake" the style. Instead of using the `<link>` tag to point to your style file(s) like you do in HTML, you can create your style file as you normally would and include it within `<style>` tags in your PHP code.

Example:

```
<style>
<?php require("style.css") ?>
</style>
```

This code snippet will effectively include the style sheets for your application. You can also use the `<fb:ref>` tag to pull the entire style sheet. Facebook also prepends your application key to the variables. The best choice for all this, of course, depends on the nature of your application, environment, and what you want to code with. d

digit

thinkdigit.com

Digit's Facebook Apps Contest

What you learn from this Fast Track could win you Rs. 20,000 in exciting prizes. Take part in Digit's Facebook Apps contest by going to

www.thinkdigit.com/d/fb_contest

4 The Basics

Before we get to the basics of creating and configuring a Facebook application, we need to set up a developer environment, learn about the basic PHP skeleton, Facebook Markup Language (FBML) and the official PHP client-libraries. We also need some information about application authorisation and authentication, Facebook sessions and signatures and the configuration settings of the Facebook Developer application.

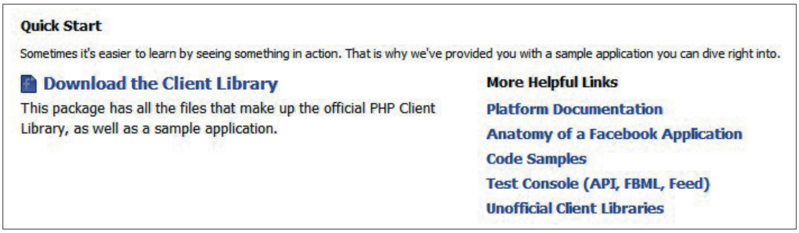
4.1 Setting a developer environment

Facebook provides the official PHP client libraries that you can download from its source code repository http://svn.facebook.com/svnroot/platform/clients/packages/facebook-platform.tar.gz and extract using any archiving tool such as WinRAR. Facebook officially supports PHP and Java client libraries, and ASP.net by its collaborated project with Microsoft - http://www.codeplex.com/FacebookToolkit. The unofficial libraries of several other programming languages can also be found on the Facebook Developers Wiki http://wiki.developers.facebook.com. This shall be your first step towards making a Facebook app. This client library contains all the resources required for your application to connect to the Facebook platform, encapsulate the standard application behaviour and access the API functions using PHP. It is nothing, but a set of three files namely,

● facebook_api_restlib.php – It lets you implement the FacebookRestClient class that provides raw HTTP access to the Facebook REST API.

● facebook.php – It lets you implement standard Facebook web-application behaviour in your application.

● facebook_desktop.php - It lets you implement standard Facebook desktop-application behaviour in your application.


Quick Start

Sometimes it's easier to learn by seeing something in action. That is why we've provided you with a sample application you can dive right into.

Download the Client Library

This package has all the files that make up the official PHP Client Library, as well as a sample application.

More Helpful Links

Platform Documentation
Anatomy of a Facebook Application
Code Samples
Test Console (API, FBML, Feed)
Unofficial Client Libraries

Download the client libraries from the same application summary page

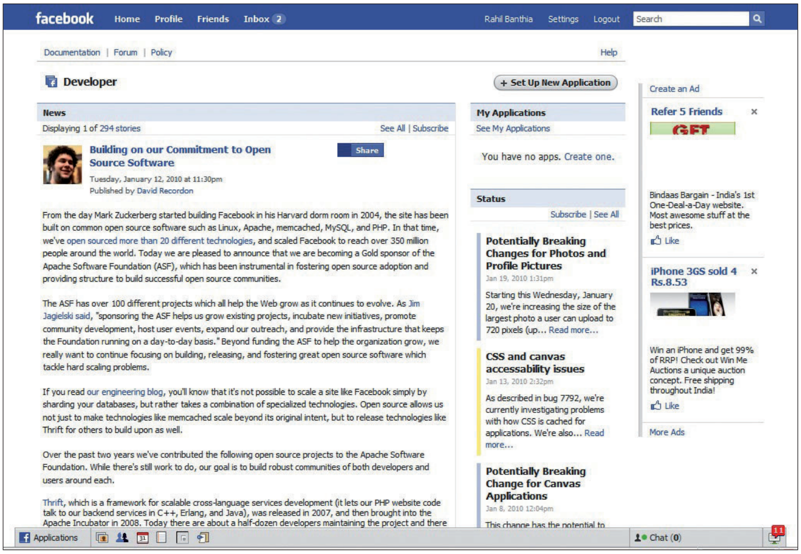
Including any of facebook.php and facebook_desktop.php enables your PHP application to direcly access the Facebook API.

Once you've the client libraries, you need to set up a web server to host any application that you create. You can either host it locally or on a web-based company that provides hosting space (There are many free solutions, but we recommend you try one of the paid servers for more security and reliability). You need to follow these steps to get your application up and running:

Before you actually start writing your application, you need to register your application with Facebook and get some application-specific information that will be essential for your application to communicate with the Facebook platform. By this, we mean the API key and the secret key that needs to be specified in your code.

Log into your Facebook account and go to www.facebook.com/developers add the application to your Facebook profile. On the top right corner, you will see the "Set up New Application" button. Click on it. Give your application a name that helps other people identify it, agree to Facebook Terms and Conditions and click on save changes.

Now the basic stuff is in place, so let us associate a logo with this application. This just makes it easy for people to recognise your application.



Building on our Commitment to Open Source Software

Tuesday, January 12, 2010 at 11:30pm
Published by David Recordon

From the day Mark Zuckerberg started building Facebook in his Harvard dorm room in 2004, the site has been built on common open source software such as Linux, Apache, memcached, MySQL, and PHP. In that time, we've open sourced more than 20 different technologies, and scaled Facebook to reach over 350 million people around the world. Today we are pleased to announce that we are becoming a Gold sponsor of the Apache Software Foundation (ASF), which has been instrumental in fostering open source adoption and providing structure to build successful open source communities.

The ASF has over 100 different projects which all help the Web grow as it continues to evolve. As Jim Jagielski said, "sponsoring the ASF helps us grow existing projects, incubate new initiatives, promote community development, host user events, expand our outreach, and provide the infrastructure that keeps the Foundation running on a day-to-day basis." Beyond funding the ASF to help the organization grow, we really want to continue focusing on building, releasing, and fostering great open source software which tackle hard scaling problems.

If you read our engineering blog, you'll know that it's not possible to scale a site like Facebook simply by sharding your databases, but rather takes a combination of specialized technologies. Open source allows us not just to make technologies like memcached scale beyond its original intent, but to release technologies like Thrift for others to build upon as well.

Over the past two years we've contributed the following open source projects to the Apache Software Foundation. While there's still work to do, our goal is to build robust communities of both developers and users around each.

Thrift, which is a framework for scalable cross-language services development (it lets our PHP website code talk to our backend services in C++, Erlang, and Java), was released in 2007, and then brought into the Apache Incubator in 2008. Today there are about a half-dozen developers maintaining the project and there

You have no apps. Create one.

Potentially Breaking Changes for Photos and Profile Pictures
Jan 19, 2010 1:31pm
Starting this Wednesday, January 20, we're increasing the size of the largest photo a user can upload to 720 pixels (up... Read more...

CSS and canvas accessability issues
Jan 13, 2010 2:32pm
As described in bug 7792, we're currently investigating problems with how CSS is cached for applications. We're also... Read more...

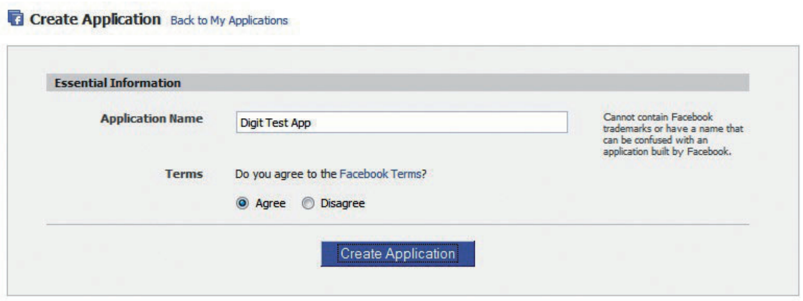
Potentially Breaking Change for Canvas Applications
Jan 8, 2010 12:04pm
This change has the potential to

Bindaas Bargain - India's 1st One-Deal-a-Day website. Most awesome stuff at the best prices.

iPhone 3GS sold 4 Rs.8.53

Win an iPhone and get 99% of RRP! Check out Win Me Auctions a unique auction concept. Free shipping throughout India!

Click on set up new application the top right corner of the developer page



Create Application Back to My Applications

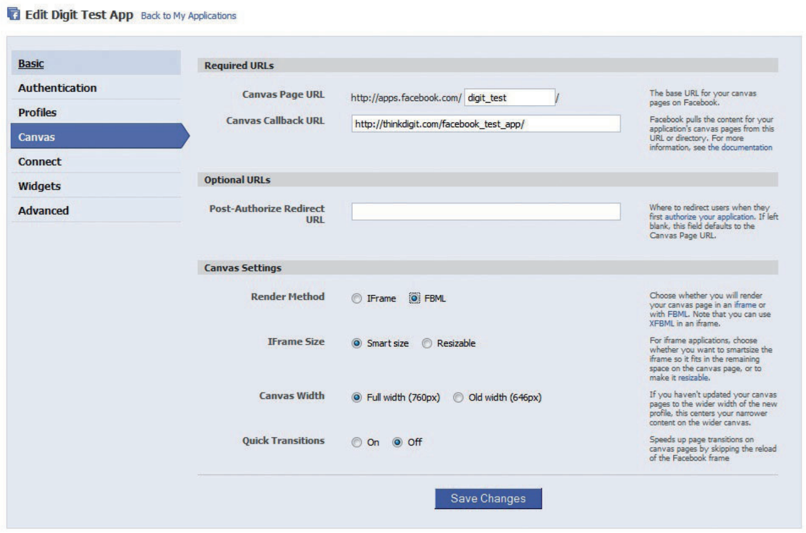
Essential Information

Application Name	Digit Test App	Cannot contain Facebook trademarks or have a name that can be confused with an application built by Facebook.
Terms	Do you agree to the Facebook Terms? [✓ Agree] Disagree	

Create Application

Enter the desired name for your application

Let us also add a description because then it becomes easy for people to find your application through the Facebook application directory. You can also add an email address that people can use for technical support.



Edit Digit Test App Back to My Applications

Basic
Authentication
Profiles
Canvas
Connect
Widgets
Advanced

Required URLs

Canvas Page URL	http://apps.facebook.com/ digit_test /	The base URL for your canvas pages on Facebook.
Canvas Callback URL	http://thinkdigit.com/facebook_test_app/	Facebook pulls the content for your application's canvas pages from this URL or directory. For more information, see the documentation

Optional URLs

Post-Authorize Redirect URL		Where to redirect users when they first authorize your application. If left blank, this field defaults to the Canvas Page URL.

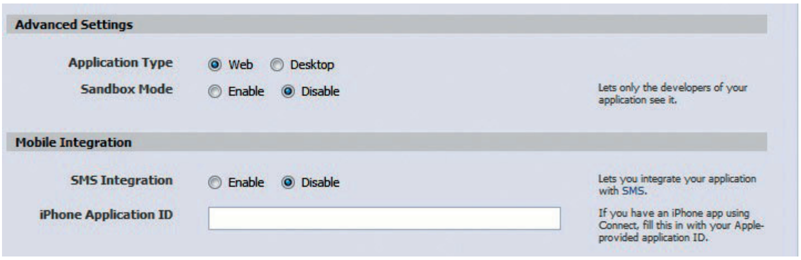
Canvas Settings

Render Method	IFrame [✓ FBML]	Choose whether you will render your canvas page in an iframe or with FBML. Note that you can use XFBML in an iframe.
IFrame Size	[✓ Smart size] Resizable	For iframe applications, choose whether you want to smartsize the iframe so it fits in the remaining space on the canvas page, or to make it resizable.
Canvas Width	[✓ Full width (760px)] Old width (646px)	If you haven't updated your canvas pages to the wider width of the new profile, this centers your narrower content on the wider canvas.
Quick Transitions	On [✓ Off]	Speeds up page transitions on canvas pages by skipping the reload of the Facebook frame

Save Changes

The canvas page of settings

Then head over straight to the canvas tab. There you will see render methods. You have to choose between iFrame and FBLM. For this example to be simple, we will use FBML. Choose it and click on save changes. You also need to populate the base and callback URL for your application. Base URL



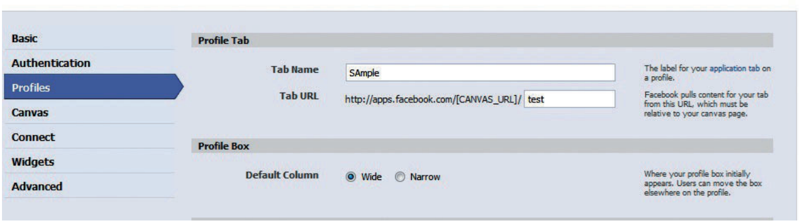
Advanced Settings

Application Type	[✓ Web] Desktop	
Sandbox Mode	Enable [✓ Disable]	Lets only the developers of your application see it.

Mobile Integration

SMS Integration	Enable [✓ Disable]	Lets you integrate your application with SMS.
iPhone Application ID		If you have an iPhone app using Connect, fill this in with your Apple-provided application ID.

Set-up the application type to web



Profile Tab

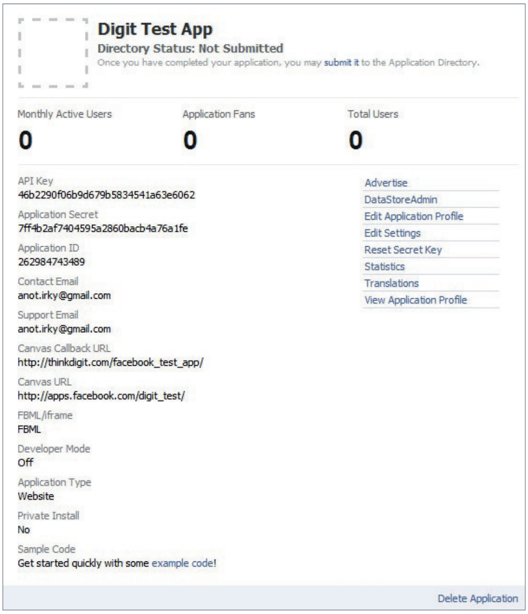
Tab Name	SAmple	The label for your application tab on a profile.
Tab URL	http://apps.facebook.com/[CANVAS_URL]/ test	Facebook pulls content for your tab from this URL, which must be relative to your canvas page.

Profile Box

Default Column	[✓ Wide] Narrow	Where your profile box initially appears. Users can move the box elsewhere on the profile.

The profiles tab

is the just the fancy URL name that other people will be using to access your application page. Facebook will provide immediate feedback if that url is unavailable, in which case you should pick something else. The callback URL is the actual stuff that points to the application to your server. Make sure you enter the URL of the folder in which your index.php and other pages are located and not the actual `index.php` URL.



Digit Test App

Directory Status: Not Submitted

Once you have completed your application, you may submit it to the Application Directory.

Monthly Active Users	Application Fans	Total Users
0	0	0

API Key
46b2290f06b9d679b5834541a63e6062

Application Secret
7ff4b2af7404595a2860bacb4a76a1fe

Application ID
262984743489

Contact Email
anot.irky@gmail.com

Support Email
anot.irky@gmail.com

Canvas Callback URL
http://thinkdigit.com/facebook_test_app/

Canvas URL
http://apps.facebook.com/digit_test/

FBML/iframe
FBML

Developer Mode
Off

Application Type
Website

Private Install
No

Sample Code
Get started quickly with some example code!

API key and secret key of your application

Next head over the Advanced settings option. Select web site for this example as we will be creating a traditional web app that integrates with Facebook and not a desktop app which accesses Facebook using the API and is run like a normal desktop application.

Now go the Authentication Options tab. Here you can select who all can add your application to their Facebook account. Specify a URL in the Post-Authorise callback URL text box. It is the URL that the Facebook sends a notification to, once a user authorizes your application. Select the wide option in the Default Profile Box column for a rich experience. Also fill in the URL that you want users to get redirected to after installing your application in the Post-Authorise redirect URL on the Canvas page. Make it your canvas URL, basically the URL you typed in the base URL field earlier.

Finally Click the Submit button. You will be taken to the My Applications page where you will get access to your API key and the secret key along with other meta information about your application.

4.1.1 Creating your Facebook Application

Now that you have your API and secret key, you are ready to create an application using the Facebook client library. We are more or less done on the Facebook site as of now. You need to uncompress the downloaded file and copy the client file libraries in the folder that you specified as your canvas URL while setting up your application. The PHP client library that we will be using is under the main folder named Facebook-platform. Next you need to create a new file named appinclude.php. You can use Dreamviewer to do this or even simple text editing softwares such as Notepad. This file will contain the Facebook initialization code required by your application to connect to the Facebook platform. You will have to include this file in all other PHP files in your application because this will contain your API key, secret key and callback URL. Type out the following code in this file:

```
  <?php
  require_once   '../facebook-platform/client/facebook.
php';
  $appapikey = '[api_key]';
  $appsecret = '[secret_key]';
  $appcallbackurl = '[web_app_callbackurl]';
  $facebook = new Facebook($appapikey, $appsecret);
```

```
$user = $facebook->require_login();
try {
if (!$facebook->api_client->users_isAppAdded()) {
$facebook->redirect($facebook->get_add_url());
}
} catch (Exception $ex) {
$facebook->set_user(null, null);
$facebook->redirect($appcallbackurl);
}
?>
```

Replace the API key, secret key and the callback URL wherever required.

Next log on to your webserver (your callback URL, remember?) and create a folder for your application. This is required because the facebook.php path in the require_once statement of the appinclude.php file we just wrote requires your app files to be located in a separate parallel folder with the Facebook client libraries. This step is flexible as you can change the path of facebook.php in appinclude.php if your folder structure differs. You are now ready to write index.php of your application. The necessary code in that file has to be

```
<h1> Digit Test Application</h1>
<?php
require_once 'appinclude.php';
?>
```

These are basically the title of your application, the PHP start and end tags and a require_once statement to include appinclude.php file.

You can play around this page as you desire. Let us make our application display the user's name. For this, you need to type in the following php code in your index.php after the necessary code `echo "<p>Your user name is: <fb:name uid=\"$user\" useyou=\"false\"/></p>";`

We are basically using the `fb:name` FBML tag to render the name of the current user. If you want to add a list of your Facebook friends, you can do this by the following code in your index.php:

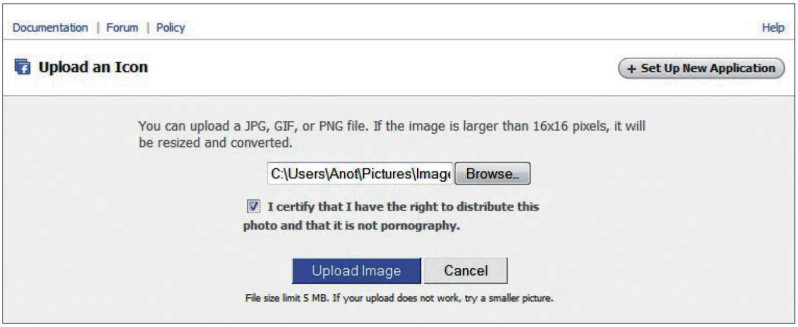
```
echo "<p>Your friends: </p>";
$friends = $facebook->api_client->friends_get();
echo "<ul>";
foreach ($friends as $friend)  {
echo  "<li><fb:name  uid=\"$friend\"  useyou=\"false\"
/></li>";
}
```

```
  echo "</ul>";
```

To try out this application, save the file, and copy it into your application folder on your web server. Next enter the canvas page URL of your application after making sure you are logged into Facebook in your web-browser. You will be redirected to your callback URL and the index.php will be rendered back inside the Facebook canvas. There's a lot more you can do with your application.

4.1.2 Add an Icon to your application

You would probably want an icon to be displayed along-side the application name. This is fairly simple. Just go back to www.facebook.com/developers, and click



Upload an Icon

You can upload a JPG, GIF, or PNG file. If the image is larger than 16x16 pixels, it will be resized and converted.

C:\Users\Anot\Pictures\Image

☑ I certify that I have the right to distribute this photo and that it is not pornography.

File size limit 5 MB. If your upload does not work, try a smaller picture.

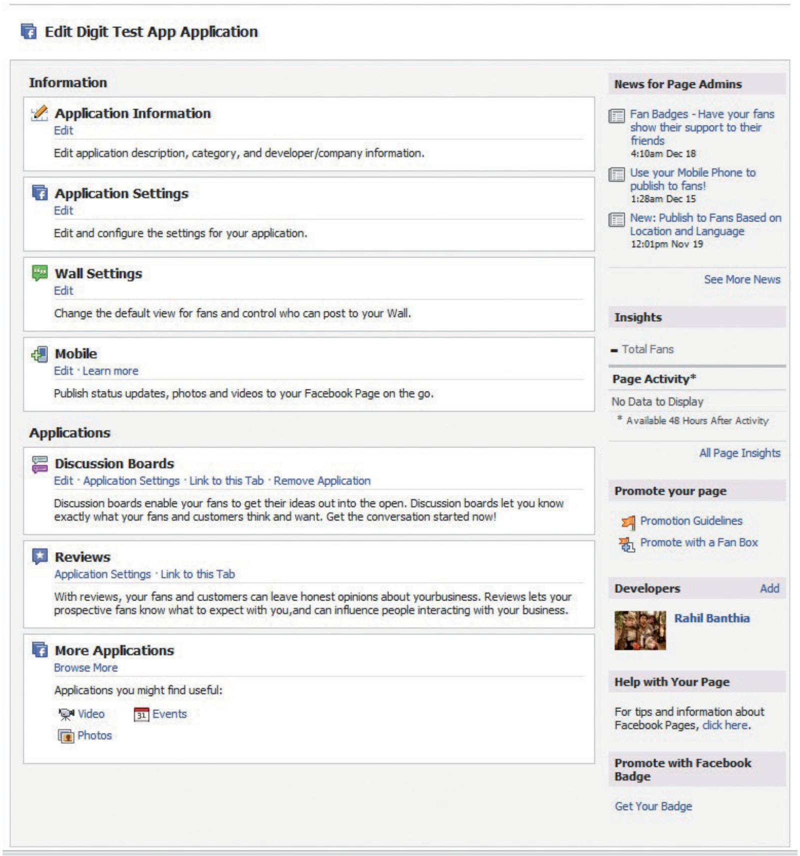
Uploading an icon for your application

on See My Apps button in the My applications box. Next, click on Edit settings. The base settings page will open. Here click on your icon link, and then on the Choose file button. Locate an image on your HDD, select the certification box and click on upload. Facebook will automatically render it and fit the required size. All you need to do now is click on save and you are done.

4.1.3 Add content to the Profile box of your application

You probably want your application to have some content when it appears as a box of the profile page. To set this up, all you need to do is add some code to the index.php file just after the require_once 'appinclude.php' statement.

```
  $default_text = "<p>Hey, <fb:name uid=\"$user\" useyou=\"false\" />! This Digit test Application is part of a sample guide to creating your own Facebook applications.</p>";
  $facebook->api_client->profile_setFBML($default_text);
```



Edit Digit Test App Application

Information

Application Information
Edit
Edit application description, category, and developer/company information.

Application Settings
Edit
Edit and configure the settings for your application.

Wall Settings
Edit
Change the default view for fans and control who can post to your Wall.

Mobile
Edit · Learn more
Publish status updates, photos and videos to your Facebook Page on the go.

Applications

Discussion Boards
Edit · Application Settings · Link to this Tab · Remove Application
Discussion boards enable your fans to get their ideas out into the open. Discussion boards let you know exactly what your fans and customers think and want. Get the conversation started now!

Reviews
Application Settings · Link to this Tab
With reviews, your fans and customers can leave honest opinions about yourbusiness. Reviews lets your prospective fans know what to expect with you,and can influence people interacting with your business.

More Applications
Browse More
Applications you might find useful:
Video Events
Photos

News for Page Admins
Fan Badges - Have your fans show their support to their friends
4:10am Dec 18
Use your Mobile Phone to publish to fans!
1:28am Dec 15
New: Publish to Fans Based on Location and Language
12:01pm Nov 19
See More News

Insights
Total Fans
Page Activity*
No Data to Display
* Available 48 Hours After Activity
All Page Insights

Promote your page
Promotion Guidelines
Promote with a Fan Box

Developers Add
Rahil Banthia

Help with Your Page
For tips and information about Facebook Pages, click here.

Promote with Facebook Badge
Get Your Badge

You can change almost anything about your application from the Application Profile page

4.1.4 Add an About Page

You also should add an about page for your application so that Facebook users get to know what your application is all about. Its also an important marketing step while you are promoting your application.

Go back to www.facebook.com/developers, and click on See My Apps button in the My applications box. Next, click on Edit Application Profile. Here is where you can also modify your Application name and the description which you initially gave while setting up your application. Add up all that's relevant to your application and click on Save.

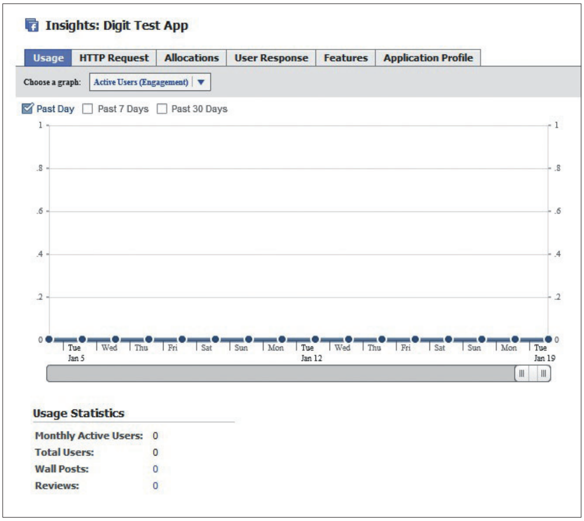
4.1.5 Redirect users

In case you want to redirect Facebook users from your app to a new URL within the canvas page, you can use the fb:redirect FBML tag. The statement required for this is-

```
<fb:redirect url="http://thinkdigit.com/" />
```

4.1.6 Viewing statics about your application

Once you are done setting up your application and creating it, you will want to view some statistics about it and how users are reacting to it. For this, go back to www.facebook.com/developers, and click on See My Apps button in the My applications box. Next, click on stats. The application stats page will appear with all the information regarding how many users used your application in the past 24 hours, total number of users who have added it, number of users who have blocked or removed it in the past 24 hours, user reviews and a summary of sampled HTTP requests along with their return statuses.



Usage Statistics

Monthly Active Users:	0
Total Users:	0
Wall Posts:	0
Reviews:	0

Detailed statistics regarding your apps usage

4.1.7 Deleting your application

If you are not satisfied with whatever work you have put in and wish to delete your application; you can do so from the My Applications page. This will also delete the API key generated for your application.

In further chapters, we will get into the finer details of App development and build the canvas page, profile boxes and give your application the ability to handle feeds, notifications, requests, photos and other multimedia. d

Win!!

Exciting Prizes worth Rs.20000

What you learn from this Fast Track could win you Rs. 20,000 in exciting prizes. Take part in Digit's Facebook Apps contest by going to

www.thinkdigit.com/d/fb_contest

5 FBML tags

FBML provides a large set of Facebook user interface and programmatic primitives required to abstract complex code and make most routine procedures effortless. The tags are automatically parsed and translated into HTML, CSS (Cascading Style Sheets), and JavaScript code by Facebook servers when a request for an application page that contains it is detected. If you've previously used HTML, then you will have no problems adapting to FBML. To distinguish between HTML and FBML commands in your application code, you need to prefix FBML tags with fb: as you would if you were using multiple DTDs/schemas in XHTML. FBML tags act differently in different locations. For example, you can use iframes on canvas pages, but you cannot use the same iframe on the code in the profile box due to platform limitations. Let us delve deeper into some well-known FBML tags.

5.1 Conditionals

FBML contains many conditional tests besides the `<fb:if>` tag that lets developers cut down on implementing conditional statements in their application. The basic `<fb:if>` tag is used like :

```
<fb:if value="true">
<p>Welcome.</p>
</fb:if>
```

The value can be set dynamically using the programming language you're making the application in. Suppose you want to test a user's ID and show a customised welcome message, you can write code like:

```
<?php
$user = $facebook->require_login();
$test = false;
if($user == 9999999){
$test = 1;
}
?>
<fb:if value="<?php echo($test)?>">
<p>This is a customized welcome message. Welcome John</
p>
<fb:else>
<p>Welcome.</p>
```

```
</fb:if>
```

You can you use the `<fb:if>/<fb:else>` conditional constructs in many more complex ways, as you will realise during the application development process. Fortunately, to reduce code, Facebook provides you will a set of tags that will do most of the common types of conditional checking:

- `<fb:is-in-network>` - test if a UID is in the specified network.
- `<fb:if-can-see>` test if the logged-in user has permissions to view the specified content. This is used to implement various privacy features in your applications.
- `<fb:if-can-see-photo>` -test if the logged-in user has permissions to view the specified photo.
- `<fb:if-is-app-user>` - test if the current user has accepted the terms and conditions of service for your application.
- `<fb:if-is-friends-with-viewer>` test if the user specified is in the friend list of the current logged-in user.
- `<fb:if-is-group-member>` - test if the current user is a member of the specified group.
- `<fb:if-is-own-profile>` - test if the viewer is the owner of the profile he/she is on currently.
- `<fb:if-is-user>` -test if the viewer is in the list of the specified users.
- `<fb:if-user-has-added-app>` test if the specified user has added and accepted the application to their account.

You should be wondering by now as to why we haven't talked about the else if FBML construct yet? It's because, there is none! If you want to perform multiple conditional checks, you will have to properly nest your if statements. Alternatively, use the FBML's switch construct. The <fb:switch> FBML tag evaluates a set of FBML tags and returns the tag information from the first tag that returns something other than an empty string. For example:

```
<?php
$user = $facebook->require_login();
$test = false;
if($user == 9999999){
$test = 1;
}
?>
<fb:switch>
<fb:if value="<?php echo($test)?>">
<fb:switch>
```

```
<fb:profile-pic uid="<?php echo($user)?>" />
<fb:default>Inner default</fb:default>
</fb:switch>
</fb:if>
<fb:default>Outer Default</fb:default>
</fb:switch>
```

5.2 User/Group information

FBML tags let your users easily interact with other users of your application. This includes handling and managing user and group information - an important part of any Facebook application.

- `<fb:name>` - This tag returns the user name for the uid passed as the parameter to the tag. It even has some customisable features and additional logic to work with users who have protected their profiles. For example, You can write `<fb:name uid="$user" ifcantsee="Anonymous">` and the tag will return "Anonymous" if the user has chosen not to display their name in their profile.
- `<fb:grouplink>` - This tag returns the name and the link of the group ID passed as parameter to the tag.
- `<fb:user>` - This tag is used to displays content to the specified user and not to everyone in general.
- `<fb:profile-pic>` - This tag renders a linked image of the user's profile picture. By default, it is a 50px x 50px image. It can be used for "iconifying" your user interactions.

5.3 Profile information

Based on where your users are accessing the application from, you can provide different content to them. This is made possible by using the various FBML tags for customising the content inside the user's profile box:

- `<fb:wide>` - This tag is used to display content only when the profile box is the wide column, i.e., the left side.
- `<fb:narrow>` - This tag is used to display the content only when the profile box is the narrow column, i.e., right side.
- `<fb:profile-action>` - This tag builds a link on the user's profile under their photo. You can use this in conjunction with setFBML to send any information to the user's profile.
- `<fb:user-table>` - This tag renders a table of the users you have specified by the `<fb:user-item>` tag.

- `<fb:user-item>` - This tag is used to define a user for use inside the `<fb:user-table>` tag.
- `<fb:subtitle>` - This tag is used to define a subtitle for the profile box.

5.4 Embedded media

FBML does not support the `<embed>` tag of HTML for embedding music, games or other rich content in your application. However, there are several tags which offer limited functionality in this are:

- `<fb:iframe>` - This tag is used to insert an iframe into your application so that you can pull in external sources to your application. This tag can only be used in the Canvas page and not in the profile page.
- `<fb:photo>` - This tag renders a picture that is in the Facebook repository if the user has permissions over it.
- `<fb:mp3>` - This tag adds a Flash-based MP3 player to your application that controls the MP3 file specified.
- `<fb:swf>` This tag adds a Flash player to render a Flash object on the Facebook pages. However, it only works on the Canvas page. On profile page, the SWF file is rendered as an image.
- `<fb:flv>` - This tag adds a Flash-based player to render and stream FLV content. .avi and other extensions are still not supported.
- `<fb:silverlight>` - This tag works in the same way as the `<fb:swf>` tag, but for Silverlight-based content.

5.5 Profile specific content

If you want to display specific content to your users depending on their profile status with your application, FBML provides you with several tags to distinguish between the application owner, users, application users (those who have granted full access to your application), and those who have added the application to their accounts:

- `<fb:visible-to-owner>` - This tag is used to display content if the user is the owner. A white space will be displayed if the user isn't the owner in place of this content.
- `<fb:visible-to-user>` - This tag is used to display content to only the specified user.
- `<fb:visible-to-app-users>` - This tag is used to display content only if the user has granted full permissions to your application.
- `<fb:visible-to-added-app-users>` - This tag is used to display content only if the user has added the application to their account.

5.6 Forms

FBML has several tags which facilitate working with form information and perform some common tasks:

- `<fb:request-form>` to create a form for sending requests. You can let your users send requests to add the application (when used with the `<fb:multi-friendseletor>` tag). This tag is invalid when working with iframes.
- `<fb:request-form-submit>` to create a submit button for use with the `<fb:request-form>` tag.
- `<fb:multi-friend-input>` to render a multifriend input field like the one used in the message composer.
- `<fb:friend-selector>` to render an autocomplete input box of the user's friends.
- `<fb:submit>` to create a JavaScript submit button for your forms. If you want to use an image instead of a submit button, you can use this like
- `<fb:submit><img src="path/to/image" /></fb:submit>`.

5.7 Others

FBML has a set of tags that help you build some very useful portions of your application:

- `<fb:comments>` to add comments to an application. You can post and redraw pages that are posted to for a given UID.
- `<fb:google-analytics>` to add JavaScript to your application so you can use Google Analytics to track your application usage.
- `<fb:mobile>` to displays content for mobile users (`http://m.facebook.com`). Content not enclosed within this tags will be ignored for mobile users.
- `<fb:random>` to randomly rotates certain parts of your application content (for quotes, pictures, or even advertising). This tag randomly choose an element from within the tag (defined by the `<fb:random-option>` tag) and also allows you to weight these options to appear more often (or less often) than other options.
- `<fb:js-string>` define a string to reference later in FBJS.
- `<fb:fbml>` define the version of FBML you wish to use.
- `<fb:fbmlversion>` to display the version of FBML that is currently being used.
- `<fb:redirect>` to redirect the browser to another URL in your application.
- `<fb:ref>` define FBML that resides at a URL that you then call later

through the tag.

- `<fb:share-button>` to create a share button with either URL information or specific metadata.
- `<fb:time>` to render the time in the user's time zone.
- `<fb:title>` to set the page's title tag.

5.8 Editor Display

FBML has a set of tags used to work with editing data. The rendered forms display in two columns with the label on the left and an input field on the right. The form the <fb:editor> tag produces uses the Post method. The different tags that can be used are:

- `<fb:editor>` The outermost tag used to create an editable form.
- `<fb:editor-button>` to create a button for your form.
- `<fb:editor-buttonset>` to create a container for one or more buttons.
- `<fb:editor-cancel>` to create a cancel button for the form.
- `<fb:editor-custom>` to insert custom FBML code, as long as it's valid FBML.
- `<fb:editor-date>` to create two select boxes in the form for the month and day.
- `<fb:editor-divider>` to add a horizontal line divider to your form.
- `<fb:editor-month>` to create a select box populated with the months of the year.
- `<fb:editor-text>` to create an input box for your form.
- `<fb:editor-textarea>` to create a textarea box for your form.
- `<fb:editor-time>` to create select boxes for hours, minutes, and an a.m./p.m. denotion for your form.

5.9 Page navigation

To build a navigation scheme for your users, the main FBML tag is the `<fb:dashboard>` tag that builds the familiar dashboard layout in Facebook. Once you do that, you can use these additional tags to manage the lay out within the dashboard, including buttons, hyperlinks, and help:

- `<fb:action>` Is used in the same way as the anchor tag in the dashboard.
- `<fb:help>` If you want to create a link to the help page of your application. This will be rendered at the right side of the dashboard.
- `<fb:create-button>` To create any button in the dashboard.
- `<fb:header>` Title Header.
- `<fb:media-header>` Media header, generally used to display user

contributed content to specific users.

- `<fb:tabs>` use it to add tabbed-navigation style links to your application. Individual tab items can be added with the `<fb:tab-item>` tag.

You can nest the <fb:action>, `<fb:help>`, and `<fb:create-button>` tags:

```
<fb:dashboard>
<fb:action href=".">Add </fb:action>
<fb:action href=".">Delete </fb:action>
<fb:help href=".">Help e</fb:help>
<fb:create-button href=".">Add </fb:create-button>
</fb:dashboard>
```

But the `<fb:tabs>` tag only allows the `<fb:tab>` tag to be nested:

```
<fb:tabs>
<fb:tab_item href="." title="Add" />
<fb:tab_item href="." title="Delete" />
<fb:tab_item href="." title="Help" />
</fb:tabs>
```

Typically you should use the <fb:tabs> tag for creating an overall navigation schema, and the <fb:dashboard> tag for performing functions within your application.

5.10 Dialog boxes

These tags provide the modal dialog boxes for your application, which is nothing but a really fancy pop-up box used to interact with your users. It is an alternative to using FBJS to create this type of interaction between your users by utilizing the Dialog Box Javascript.

`<fb:dialog>` Creates the container for the dialog box.

`<fb:dialog-title>` Set the title for your dialog box.

`<fb:dialog-content>` Used to write the message contained in the dialog box.

`<fb:dialog-button>` Used to renders button for your dialog box.

The FBML snippet for constructing a dialog box goes like:

```
<fb:dialog id="fb_test">
<fb:dialog-title>Sample Dialog Box</fb:dialog-title>
<fb:dialog-content>Content</fb:dialog-content>
<fb:dialog-button type="button" value="Okay" close_
dialog="1" />
```

```
  </fb:dialog>
  <a href="" clicktoshowdialog="fb_est">show fb:dialog</
a>
```

Within the `<fb:dialog-content>` tag, you can add other information (and other FBML) tags, such as forms to perform more advanced interactions with your users.

To generate a search form to search Faceboo, use the following code snippet:

```
  <fb:dialog id="fb_search" cancel_button="true">
  <fb:dialog-title>Search Facebook</fb:dialog-title>
  <fb:dialog-content>
  <form      action="http://www.facebook.com/s.php"
method="get">
  <input type="text" name="q" />
  <input type="submit" value="Search" />
  </form>
  </fb:dialog-content>
  </fb:dialog>
  <a href="" clicktoshowdialog="fb_search">Show Search</
a>
```

If you are not a JavaScript expert, the <fb:dialog> tags will provide a very convenient method to do most of the same things you can do with FBJS without writing any FBJS.

5.11 Wall

FBML tags which add the functionality to your application which concerns with users interacting with their wall are

- `<fb:wall>` Your application will render a wall-like section that has the `<fb:wallpost>` elements from your application users.
- `<fb:wallpost>` Used to handle the message for the wall post that can contain an `<fb:wallpostaction>` element.
- `<fb:wallpost-action>` Used to add a link at the bottom of the wall post.

You just need to make a new table with three tuples (columns) to hold the UID (bigint), the actual post (text), and a time stamp (for indexing) to handle wall posts. Additionally, you can add a primary key field, though the time stamp should suffice for this. Now, loop over these results to provide them in the `<fb:wallpost>` tags, and wrap it in `<fb:wall>`. d

6 FQL and FJS

JavaScript is essential for developing enriched user interfaces. As a result, Facebook came out with its own version of JavaScript called Facebook JavaScript.

6.1 FJS essentials

Facebook takes the FBJS, parses the code, and prepends functions and variable names with your application identifier.

For example:

```
function say_hello(name){
var my_string = 'Hello World';
return my_string + ' ' + name;
}
is converted into:
function <app_id>_say_hello(<app_id>_name){
var <app_id>_my_string = 'Hello World';
return <app_id>_my_string + <app_id>_name;
}
```

You also need to keep a few things in mind when you are using FBJS in your applications. Your scripts will perform differently in different areas of the application.

For example, let's see how the following example from the Facebook wiki behaves:

```
<a href="#" id="hello" onclick="hello_world(this);
return false;">Hello World!</a>
<script>
<!--
function random_int(lo, hi) {
return Math.floor((Math.random() * (hi - lo)) + lo);
}
function hello_world(obj) {
var r = random_int(0, 255);
var b = random_int(0, 255);
var g = random_int(0, 255);
var color = r+', '+g+', '+b;
obj.setStyle('color', 'rgb('+color+')');
}
hello_world(document.getElementById('hello'));
//-->
</script>
```

This program randomly sets the colour of the anchor text “Hello World”. However, it will behave differently depending on where your application is located. In the profile box, inline scripts like these are deferred until the first “active” event is triggered (for example, anything that requires a mouse click). But, it works perfectly fine in on the Canvas page.

6.1.1 Creating a Drop-Down Menu in FBJS

You can create Drop down menus in your Facebook application with the help of FBML and FBJS. Here’s how:

```
<div style="padding:20px">
<div class="nav" >
<ul>
<li style="width:70px;" id="home" >
<div><a onclick='displayMenu("homechild")' href='#'>Home
</a></div>
</li>
<li style="width:110px;" >
<div><a        onclick='displayMenu("abchild")'
href='#'>Addressbook</a></div>
<ul id="abchild" style="background-color:blue">
<li><a href="#">Stories</a></li>
<li><a href="#">Articles</a></li>
<li><a href="#">Game REviews</a></li>
</ul>
</li>
<li style="width:110px;" >
<div><a onclick='displayMenu("helpchild")' href='#'>Help
</a></div>
<ul id="helpchild" style="background-color:blue" >
<li><a href="#">Topics</a></li>
<li><a href="#">Contact us</a></li>
</ul>
</li>
</ul>
```

```
</div>
<div style="margin-top:140px" ></div>
</div>
```

Next, add this stylesheet to make them display properly:

```
<style>
.nav ul
{
width: 100%;
list-style: none;
line-height: 1;
height:60px;
font-weight: bold;
padding: 0px;
border-width: 1px 1px;
margin: 0 0 0 0;
}
.nav li {
float: left;
padding: 0px;
padding-top:25px;
margin-bottom:10px;
}
.nav li ul
{
display:none;
padding:5px;
padding-left:0px;
background-color:transparent;
margin-top:10px;
position: absolute;
height: auto;
width: 100px;
font-weight: normal;
border-width: 0.25em;
margin: 0px;
}
.nav li ul li{
float:none;
padding-top: 5px;
padding-bottom:5px;
padding-left:15px;
color:white;
}
.nav a:hover{
text-decoration:underline;
}
.nav li ul li a
{
color:white;
text-decoration:none;
}
</style>
```

And finally, write the following FBJS code:

```
<script>
function displayMenu(ch)
{
document.getElementById("abchild").
setStyle('display','none');
document.getElementById("helpchild").
setStyle('display','none');
try{
document.getElementById(ch).
setStyle('display','block');
}
catch(e){}
}
</script>
```

6.1.2 Dialog

If you are not content with the Dialog box creation abilities of FBML, you can do so in FBJS also very easily. Here's the code required to create a Dialog object: `Var dialog = new Dialog();`

To display information within your Dialog box, Use the showMessage() method of a Dialog along with its three parameters:

```
<script>
new Dialog().showMessage("Hello World","Are you a
G33|<","Boring!");
  </script>
```

To accept User input using FBJS, you need to use the showChoice() method of the Dialog object along with its four parameters:

```
  dialog.showChoice('Wheee', 'mmmmm' ,':-/','Loser');
```

To manage the user input, after they click Okay or Cancel:

```
  dialog.onconfirm    =    function(){new    Dialog().
showMessage("Info","OK")};
  dialog.oncancel    =    function(){new    Dialog().
showMessage("Info","Cancel")};
```

So, the final script will be like this:

```
  <script>
  var dialog = new Dialog();
  dialog.showChoice('Wheee', 'mmmmm' ,':-/','Loser');
  dialog.onconfirm    =    function(){new    Dialog().
showMessage("Info","OK")};
  dialog.oncancel    =    function(){new    Dialog().
showMessage("Info","Cancel")};
  </script>
```

In FBJS, you can display two types of dialog box: popup, and contextual. For a contextual dialog, use the following code:

```
  <div style="padding:20px">
  <a id="link" style='cursor:pointer' onclick="showDialog
(this);">Click to see a contextual dialog box</a>
  </div>
  <script>
  function showDialog(linker){
  //context = document.getElementById("link");
  new Dialog(Dialog.DIALOG_CONTEXTUAL).setContext(linker).
showMessage("HelloWorld");
  }
  </script>
```

You need to call setContext() before calling showMessage() or showChoice(). If you call it later, the dialog will not be shown in the right place, instead it will go to the top left corner of the page.

6.1.3 Advanced input

Apart from a Boolean input like Okay or Cancel, you can take multiple from your users and even pass any valid pre-rendered FBML block to dialogs. The following example shows how to take written inputs from your users.

```
  <div style="padding:20px">
  <a id="link" href="#" onclick="showDialog(this);">Click
to see a contextual dialog box</a>
  </div>
  <fb:js-string var='inputs'>
  Wassaa?? <br/>
  <input type="text" id="userinput" />
  </fb:js-string>
  <script>
  function showDialog(linker){
  //context = document.getElementById("link");
  New          Dialog(Dialog.DIALOG_CONTEXTUAL).
setContext(linker).showMessage("Greeting",inputs,"Okay").
onconfirm=function(){
  data    =    "So,    you're    upto    "    +    document.
getElementById("userinput").getValue()   +"Pretty   cool
man!";
  new Dialog().showMessage("Info",data,"OK");
  return false;
  };
  }
  </script>
```

6.1.4 Events

Managing events in FBJS is pretty simple if you are familiar with event management in JavaScript because its almost is the same. You can use the addEventListener () function to attach listeners to an event. It takes two arguments; the name of the event to listen, and the callback.

Let's try this with an example:

```
  <div style="padding:20px">
  <div id="somediv">
  Your Content
  </div>
  <input type="button" id="somebutton" value="Click Me" />
  </div>
  <script>
  button = document.getElementById("somebutton");
  button.addEventListener("click",hide);
  function hide(event)
  {
  document.getElementById("somediv").
setStyle("display","none");
  }
  </script>
```

To find out who fired the event, use the event.target.getId() method of an event object.

```
  function hide(event)
  {
  document.getElementById("somediv").
setStyle("display","none");
  usedby = event.target.getId();
  new Dialog().showMessage("Info","This event was used by
"+usedby+"","Okay")
  }
```

When you attach an event to a Data Object Model node, the event gets automatically added to all the child nodes in that node. So just which object is invoking the event listener, and then take action using getId().

There are other useful methods available to this event object, which are essential in developing sophisticated event-based applications:

- `stopPropagation()`—Stop the propagation of the event further to any DOM sub- node.
- `cancelDefault()`—Cancel the default behaviour of an event.
- `listEventListeners(eventName)`—Return an array of handles of the events which have been added to this event. It also includes the events that were added in FBML using the <event> attribute.
- `purgeEventListeners(eventName)`—Remove all event listeners for this event.

Example:

```
  <div style="padding:20px">
  <textarea id="sometextarea">
  </textarea>
  </div>
  <script>
  ta = document.getElementById("sometextarea");
```

```
ta.addEventListener("keydown",keydown);
function keydown(ev)
{
keycode = ev.keyCode;
if (keycode==83)
ev.preventDefault();
//new Dialog().showMessage("Info","you just pressed
"+keycode+"","OK")
}
</script>
```

6.2 FQL essentials

FQL uses the same syntax as typical ANSI-SQL, so if you've worked with SQL before (and I assume you have), FQL isn't a big deal. There are only nine tables to deal with and no joins. FQL does not implement the LIMIT, GROUP BY, or ORDER BY clauses that are common to ANSI SQL–compliant implementations. Let's take a look at the tables and fields that are part of the Facebook Query Language.

6.2.1 Tables

Here's a list of different tables or views of specific data to make sure you know what's available to you.

- `users(uid, first_name, last_name, pic_small, pic_big, pic_square, affiliations, profile_update_time, timezone, birthday, sex, hometown_location, meeting_sex, meeting_for, relationship_status, significant_other_id, political, current_location, activities, interests, is_app_user, music, tv, movies, books, quotes, about_me, hs_info, education_history, work_history, notes_count, wall_count, status, has_added_app)`
- `friend(uid1, uid2)`
- `group(gid, name, pic_small, pic_big, pic, description, group_type, group_subtype, recent_news, creator, update_time, office, website, venue)`
- `group_member(uid, gid)`
- `event(eid, name, tagline, pic_small, pic_big, pic, host, event_type, event_subtype, start_time, end_time, creator, update_time, location, venue)`

- event_member(uid, eid, rsvp_status)
- photo(pid, aid, owner, src_small, src_big, src, caption, created)
- album(aid, cover_pid, owner, name, created, description, location, size)
- photo_tag(pid, subject, xcoord, ycoord)

6.1.3 Functions and operators

FQL has several simple operators and functions to help you work with user data. There are Boolean operators (AND, OR, and NOT), comparison operators (=, >, >=, <, <=, <>), and arithmetic operators (+, -, *, and /). The functions included in FQL also allow you to perform some basic string manipulations, although not very exhaustively.

- now() – This returns the current time.
- strlen(string) - return the length of the string passed to the function.
- concat(string1, string2,…, stringn) - concatenate n strings together.
- substr(string, start, length) - return a substring from a given string.
- strpos(haystack, needle) - return the position of the character needle in the string haystack.
- lower(string) - cast the given string to lowercase.

upper(string) - cast the given string to uppercase.

Example:

```
<?php
$facebook_config['debug'] = false;
$facebook_config['api_key'] = '<your_api_key>';
$facebook_config['secret_key'] = '<your_secret_key>';
require_once('<path_to_client_library>/facebook.php');
$facebook = new Facebook($facebook_config['api_key'],
$facebook_config['secret']);
$user = $facebook->require_login();
$fql = "SELECT concat(first_name, ' ', last_name)
FROM user
WHERE uid = '$user'";
$fql_result = $facebook->api_client->fql_query($fql);
echo "<pre>FQL Result:" . print_r($fql_result, true) .
```

```
"</pre>";
  ?>
```

This simply selects the first and last names of the user who is currently making the request. The result will be an array in the following format:

```
  FQL Result:Array
  (
  [0] => Array
  (
  [anon] => <your_first_name> <your_last_name>
  )
  )
```

We've used the concat() function to group several tuples together to conserve bandwidth. You can also use this to put a specific string into your page that combines several fields in a specific way. Letting the Facebook servers do some of this processing before it gets back to your server will cut down on your bandwidth in order to speed up your application and decrease your server load. You can also perform various subqueries. For example, FQL equivalent for the facebook.events.get REST API call goes like:

```
  SELECT eid, name, tagline, nid, pic, pic_big, pic_small,
host, description, event_type, event_subtype, start_time,
end_time, creator, update_time, location, venue
  FROM event
  WHERE eid IN (SELECT eid FROM event_member
  WHERE uid=uid AND rsvp_status=rsvp_status) AND eid IN
(eids) AND end_time >= start_time AND start_time < end_
time
```

Nested queries like these are very useful for testing set membership, set comparisons, and set cardinality. You will however need to take some additional processing to make sure your information is displayed in a specific order in your client language like PHP. If you want to sort the array and then slice it:

```
  <?php
  $fql = "SELECT eid, name, location
  FROM event
  WHERE eid IN (
  SELECT eid
  FROM event_member
  WHERE uid = '$user'
```

```
)";
$fql_result = $facebook->api_client->fql_query($fql);
asort($fql_result);
array_slice($fql_result, 0, 4);
?>
```

An FQL query is passed to find events for the current user, sort the resulting PHP array, and returns the array with the five elements (positions 0–4) of the query result.

FQL gives developers granular control of the information about your users from the Facebook servers. As the complexity of your FQL increases with subqueries, you might run into problems. For this, you can use the Facebook API Test Console at http://developer.facebook.com/tools.php to debug your code. This tool provides invaluable help in discerning where your bugs exist when you are working with more complicated interactions with FBML and FQL. d

digit

thinkdigit.com

Digit's
Facebook Apps
Contest

What you learn from this Fast Track could win you Rs. 20,000 in exciting prizes. Take part in Digit's Facebook Apps contest by going to

www.thinkdigit.com/d/fb_contest

7 Updating profile pages

The profile page of Facebook provides a snapshot of who a user is and what he/she finds important. A Facebook application can add profile boxes, application tabs and info sections to manipulate this information.

7.1 Profile boxes

The Profile boxes are FBML based and do not allow profiles to use IFrames or Flash. The content is set by the application using the profile.setFBML() API call. There are three types of Profile Boxes in Facebook, namely Main, Wide and Narrow. The Main box appears on the left side of the Wall and the Info tab of a profile, and measures 184x250 pixels. A maximum of four of these appear at a time on the Profile page, and the rest are moved to the Boxes tab. The Narrow and Wide boxes appear on the Boxes tab by default. Wide appears on the left side and is 380 px wide, whereas the Narrow appears on the right and is 184px wide. Both of these don't have any height limitations. These profile boxes can be moved by the user by dragging and dropping, but the content inside a `<fb:narrow>` tagged box only shows up when the profile box is on the right side, while content in a `<fb:wide>` tagged box only shows up on the left side. A good profile box will support both the locations by displaying the content formatted appropriately based on size limitations.

The PHP client library has different parameters for the `profile_setFBML()` call:

Parameter	Details
markup	It has been deprecated. Just pass it as NULL.
Uid	This denotes the user ID for the user or Public Profile to update, which is set to the current user by default.
profile	It is a string parameter which contains the FBML for the profile box on the Boxes tab.
profile_action	It has been deprecated. Just pass it as NULL.
mobile_profile	It is a string parameter which contains the FBML that appears on http://m.facebook.com.
profile_main	It is a string parameter which contains the FBML that appears on the Main profile box.

Example:

```
$markup = NULL;
$mainFBML = "Appears on the Wall";
```

```
  $profileFBML = "
  <fb:narrow>Appears on right side</fb:narrow>
  <fb:wide>Appears on left side</fb:wide>";
  $mobileFBML = "
  <fb:mobile>Appears on m.facebook.com</fb:mobile>";
  $profileAction = NULL;
  $facebook = new Facebook(FB_API_KEY, FB_APP_SECRET);
  $facebook->api_client->profile_setFBML($markup,
$appUserID, $profileFBML,
  $profileAction, $mobileFBML, $mainFBML);
```

7.2 Creating the Profile Box

We will update our sample application to use all the profile boxes in this section. These profile boxes lets you send feedback. There is a Send button that takes the user back to the application Canvas page.

Add a call to the `updateProfileBox()` function in index.php and index_iframe.php by inserting the command `updateProfileBox( $facebook->user );` just before `renderPage():`

Create a File called profile.inc inside the inc directory on your server. This file will contain all the code related to creating profile content and updating it on Facebook. You also need to add `require_once 'inc/profile.inc';` to the top of your index.php and index_iframe.php files. The following source code goes into the profile.inc file:

```
  <?php
  function updateProfileBox($appUserID){
  $facebook = new Facebook(FB_API_KEY, FB_APP_SECRET);
  $generatedProfileFBML = generateProfileBoxFBML($appUse
rID);
  $profileFBML = $generatedProfileFBML['profile'];
  $mainFBML = $generatedProfileFBML['main'];
  try {
  $facebook->api_client->profile_setFBML(NULL, $appUserID,
$profileFBML, NULL, NULL, $mainFBML);
  }    catch (Exception $ex) {
  wr("<Exception: uid: $appUserID - " . $ex->getMessage()
. ">" );
  return false;
  }
  return true;
  }

  function generateProfileBoxFBML($appUserID) {
  global $g_categories;
  $styles   =   htmlentities(file_get_contents(LOCAL_APP_
DIR.'/css/profile.css'),
  true);
  $headerBlock = "
  <style>
  $styles
  </style>
  <fb:subtitle>
  <fb:action   href='".FB_APP_URL."'>Send   Feedback</
fb:action>
  </fb:subtitle>";
  $categories = "";
  foreach( $g_categories as $name => $info ){
  $categories .="
  <div class='category'>
  <img class='categoryImg' title='{$info['title']}'
  src='".LOCAL_APP_URL."/img/".$info['smallimg']."'/><br>
  <div class='categoryTitle'>{$info['title']}</div>
  <input type='radio' name='category' value='$name' />
  </div>";
  }
  $formBlock = "
  <div class='sendFeedback'>
  <span class='sendTitle'>Send Feedback</span>
  <form method='POST' id='FeedbackForm' action='".FB_APP_
URL."/' >
  <div class='categories'>
  $categories
  <div style='clear:both;'></div>
  </div>
  <input type='hidden' name='target' value='$appUserID'/>
  <input class='textInput' name='Feedback'/><br/>
  <input      class='inputbutton'      type='submit'
```

```
name='submitFeedback'
  value='Send'>
  </form>
  </div>";
  $wideContent = "
  <fb:wide>
  <div class='wide'>
  <div class='profileContent'>
  <img class='banner' src='".LOCAL_APP_URL."/img/banner_w.
png'/>
  $formBlock
  </div>
  </div>
  </fb:wide>";
  $narrowContent = "
  <fb:narrow>
  <div class='narrow'>
  <div class='profileContent'>
  <img class='banner' src='".LOCAL_APP_URL."/img/banner_n.
png'/>
  $formBlock
  </div>
  </div>
  </fb:narrow>";
  $mainContent = "
  <div class='main'>
  <div class='profileContent'>
  <img class='banner' src='".LOCAL_APP_URL."/img/banner_n.
png'/>
  $formBlock
  </div>
  </div>";
  $profileText['profile'] = $headerBlock . $wideContent .
$narrowContent;
  $profileText['main'] = $headerBlock . $mainContent;
  return $profileText;
  }
  ?>
```

7.3 Styling the Profile Boxes

To add style for the profile boxes you create using the above methods, create a new file called profile.css in the CSS directory and add the following source code.

```
  .profileContent { margin-top:-12px; text-align:center;}
  .sendFeedback { text-align:center;}
  .sendTitle {font-weight: bold;}
  #FeedbackTable .textInput {width: 200px;}
  .categories { padding-bottom: 5px; }
  .categoryTitle { height:30px; font-weight:bold; }
  .category { float:left; text-align:center; margin:3px; 
width:43px;
  height:65px; padding:5px; }
  .categoryImg { width:24px; height:24px;}
  .wide .banner { width:380px; height:45px; }
  .wide .category { width:108px; }
  .wide .categoryTitle { height:20px;}
  .narrow .banner { width:193px; height:23px; }
  .main .banner { width:193px; height:23px;}
```

7.4 Adding the Profile Boxes

Profile boxes do not get automatically added to a user profile. The user must manually choose to add one to his profile by clicking on a button.

7.4.1 Displaying Add to Profile Buttons in FBML Canvas Pages

Displaying the Add to Profile button is very simple for an FBML canvas page. Just add the following code to the renderPage() function in index.php, after the banner in the header block. The `<fb:if-section-not-added section='profile'>` tag makes sure Facebook only renders the inside FBML if the user hasn't added the profile box yet. The `<fb:addsection-button section='profile' />` tag actually renders the button. All of this is enclosed in a <div> so that we can position it where we want on the page using CSS.

```
  index.php
  <div class='banner'
  style='background:  url(".LOCAL_APP_URL."/img/banner.
png) no-repeat;' >
  <div id='buttons' class='clearfix' >
```

```
<div id='addbutton'>
<fb:if-section-not-added section='profile'>
<fb:add-section-button section='profile' />
</fb:if-section-not-added>
</div>
</div>
</div>
```

7.4.2 Displaying Add to Profile Buttons in IFrame Canvas Pages

Adding the Add to Profile button in an IFrame Canvas Page is slightly more tricky because there is no way to create this button without using FBML and IFrames don't support FBML. We need to use XFBML for this, which requires the Facebook JavaScript Client Library and all the supporting code.

The first step in this is to update the Developer Settings for this application. Browse to www.facebook.com/developers/apps.php and click the Edit Settings link for Feedbacks. Click on the Connect tab and enter in your Canvas Callback URL into the Connect URL field and click on Save. Next, create a file named xdreceiver.html in your application root directory, and add the following source code into it.

```
xdreceiver.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>Cross-Domain Receiver Page</title>
</head>
<body>
<script src="http://static.ak.facebook.com/js/api_lib/
v0.4/XdCommReceiver.js?2"
type="text/javascript"></script>
</body>
</html>
```

The Facebook JavaScript Client Library will now be able communicate with your application. Next create a placeholder <div> that the Library can use. The renderPage() function in index_iframe.php should now look like:

```
<div class='banner'
style='background:  url(".LOCAL_APP_URL."/img/banner.
png) no-repeat;' >
```

```
  <div id='buttons' class='clearfix' >
  <div id='addbutton'></div>
  </div>
  </div>
```

Next add a script block that loads and initializes the Facebook JavaScript Client Library and tells it to replace the placeholder `<div>` with the Add to Profile button. Use the following code at the end of renderPage() in index_iframe.php , just before the `</body>` tag to accomplish this.

```
  index_iframe.php
  $pageOutput .= "
  <script
  src='http://static.ak.facebook.com/js/api_lib/v0.4/
FeatureLoader.js.php'
  type='text/javascript'></script>
  <script type='text/javascript'>
  FB_RequireFeatures(['XFBML'], function() {
  FB.Facebook.init('".FB_API_KEY."',  'xdreceiver.html',
null);
  FB.Connect.showAddSectionButton('profile',
  document.getElementById('addbutton'));
  FB.Connect.showAddSectionButton('info',
  document.getElementById('infobutton'));
  });
  </script>
  </body>
  </html>
  ";
```

As with all the XFBML, this code runs on the client, so there might be a slight delay as the library needs to render the new button HTML. To test the profile boxes, go to your application and click on the Add to Profile button on either the FBML or IFrame tab. Click on Add, and go to your profile. Click on Keep to accept the new profile box. You can try moving it from the Wall to the Boxes tab and from the Narrow to the Wide side to see how the layout changes with each location.

7.5 Application tabs

If you want your applications to display profile content in a much larger space, you have to use Application tabs. They behave like profile boxes, but are 760px wide. Application tabs are different in the sense that they are populated via a callback instead of presetting the content once via `profile.setFBML()`.

If an application supports an application tab, the user can click on the + at the right side of his profile tab list and select it from a list to add the tab. You need to update the Developer Settings to allow users to add an application tab by going to the Profiles tab and filling in Feedbacks for the Tab Name and apptab.php for the Tab URL. Next, you need to create a new file named apptab.php in the application root directory and add the following source code:

```
  <?php
  require_once dirname(__FILE__).'/inc/globals.inc';
  require_once dirname(__FILE__).'/inc/db.inc';
  require_once dirname(__FILE__).'/inc/utils.inc';
  require_once dirname(__FILE__).'/inc/profile.inc';
  dumpRequestVars(true, basename(__FILE__));
  $facebook = new Facebook(FB_API_KEY, FB_APP_SECRET);
  $generatedProfileFBML=generateProfileBoxFBML($facebook-
>profile_user);
  $appTabFBML = $generatedProfileFBML['tab'];
  echo $appTabFBML;
  ?>
```

This file acts as the handler for the application tab callback and reuses functionality from profile.inc to help generate the FBML. The code calls the generateProfileBoxFBML() function and gets the FBML for the tab and then echoes it out. Update profile.inc with the following source code to add the tab specific content.

```
  $appTabHeaderBlock = "
  <style>
  $styles
  </style>
  <div class='subtitle'>
  <a href='".FB_APP_URL."'>
  Send a Feedback
  </a>
  </div>";
  $tabContent = "
  <div class='tab'>
  <div class='profileContent'>
  <img class='banner' src='".LOCAL_APP_URL."/img/banner.
png'/>
  $formBlock
  </div>
  </div>";
  $profileText['tab'] = $appTabHeaderBlock . $tabContent;
```

Add the `$appTabHeaderBlock` section just after the `$headerBlock` in your file and the `$tabContent` section after `$mainContent`.

Note that application tabs do not support the `<fb:subtitle>` tag we used for the profile boxes, so we will have to manually create the link. Just like the other profile boxes, enclose the FBML in a `<div>` tag with the class tab so that you can apply custom styles.

```
  profile.css file needs to be updated with:
  .tab .banner { width:760px; height:90px; }
  .tab .sendTitle {font-size:20px; }
  .tab .category { width:108px; }
  .tab .categoryTitle { height:15px; }
  .tab .textInput { width: 400px; }
  .tab .profileContent { text-align:right; margin:0px;}
  .subtitle { text-align:right; border-bottom:1px solid
#ECEFF5; color:#444444;
  margin:0px 0px 5px 0px; padding:2px 8px; }
```

7.6 Creating info sections

Add a call to a new function, `updateInfoSection()` in your index.php and index_iframe.php files, just after `updateProfileBox()`:

```
  updateInfoSection( $facebook->user );
```

Create a new file called info.inc in the inc directory, and add the following source code:

```
  <?php
  function updateInfoSection($appUserID) {
  global $g_categories;
  $db = new DB();
  $infoData = $db->getFeedbacksForUser($appUserID);
  $facebook = new Facebook(FB_API_KEY, FB_APP_SECRET);
  $uids = array_keys($infoData);
  $uids[] = $facebook->user;
  $users = array();
  $userNames = $facebook->api_client->users_getInfo($uids,
'name');
  foreach ($userNames as $user) {
  $users[$user['uid']] = $user['name'];
  }
  $FeedbackInfoFields = array();
  foreach( $infoData as $Feedback ) {
  $FeedbackInfoFields[] = array(
  'label'=> $users[$Feedback["targetID"]].' is '.
  $g_categories[$Feedback["category"]]["title"],
  'image' => LOCAL_APP_URL."/img/".
  $g_categories[$Feedback["category"]]["bigimg"],
  'sublabel' => 'Sent by '.$users[$Feedback["appUserID"]],
  'description'=> 'because '.$Feedback["Feedback"],
  'link'=> FB_APP_URL );
  }
  $infoFields = array(
  array('field' => 'Feedbacks Received', 'items' =>
$FeedbackInfoFields)
  );
  $facebook->api_client->profile_setInfo( 'Feedbacks', 5,
$infoFields,
  $appUserID );
  }
  ?>
```

This file now contains all the code needed to create the content for our info sections and update them on Facebook. Next, add this statement to the top of index.php and index_iframe.php files to include this new file: require_once 'inc/info.inc';

Just like the Add to Profile buttons we previously added, we need to update our canvas pages to display the Facebook-provided Add to Info button. If its an FBML canvas page, add the following code under the `addbutton <div>` in index.php:

```
  <div id='infobutton'>
  <fb:if-section-not-added section='info'>
  <fb:add-section-button section='info' />
  </fb:if-section-not-added>
```

```
</div>
```

If its an IFrame, add the following line for the new infobutton placeholder <div> after the addbutton placeholder <div> in index_iframe.php:

```
<div id='infobutton'></div>
```

Also, add this line right after the showAddSectionButton() call for the Profile button in index_iframe.php:

```
FB.Connect.showAddSectionButton('info',document.
getElementById('infobutton'));
```

Finally, you need to add the following line to main.css to position the button in the banner:

```
#infobutton { position:absolute; top:48px; right:33px; }
```

7.7 Managing the FBML Cache

FBML for profile boxes are cached. This works fine with applications with completely static content. But you might want to update a user's profile boxes when he/she accesses the application to reflect the most current information. If you want to update many of your users' profile boxes at once because of a general style change or content change that applies to a large set of users, you can simply to set up a server-side cron job to iterate through all the application's users and call `profile.setFBML()` for them. However, if your application has a large number of users, this can take forever to accomplish, and be unmanageable. For this very reason, Facebook provides you the `<fb:ref>` FBML tag can mark blocks of FBML that can be updated across many profiles with a single call. There are two ways in which you can use this tag: by URL and by Handle. They both eventually achieve the same result, but one might fit better with your application design than the other. If you specify a URL, Facebook calls that URL to get the new content when the cache is being refreshed, while if you specify a Handle, your application directly passes the new FBML content. The `<fb:ref>` tags can also be nested, but only the content of the Handle or URL specified is updated.

Let us see how an application might use `<fb:ref>` with a Handle. Create a block of FBML and pass it to `fbml.setRefHandle()` with a unique Handle identifier. Then, use `<fb:ref handle='youridentifier'/>` in your profile FBML content wherever you want that block of FBML to appear and pass it to `profile.setFBML()`. After this, your application can just call `fbml.setRefHandle()` with new FBML to update all profiles that contain that Handle to use the new content. Only the section in the Handle will be updated.

```
  $commonFBML = "<div>This month's theme is xxxxx</div>";
  $facebook->api_client->fbml_setRefHandle("temp",
$commonFBML);
  $profileFBML = "
  <div>
  <h1>Theme</h1>
  <fb:ref handle='temp'/>
  </div>";
  $facebook->api_client->profile_setFBML(NULL, $appUserID,
$profileFBML,
  NULL, NULL, NULL);
```

You can also do the same thing by using a URL and not setting the cache the FBML ahead of time. Instead, supply a callback URL from which the FBML can be retrieved. Call fbml.refreshRefUrl() and pass the URL to update it. The callback will return the FBML, and will only be called one time, not once per profile updated.

```
  $profileFBML = "
  <div>
  <h1>This month's theme</h1>
  <fb:ref url='http://thinkdigit.com/theme/>
  </div>";
  $facebook->api_client->profile_setFBML(NULL, $appUserID,
$profileFBML,
  NULL, NULL, NULL);
  $facebook->api_client->fbml_refreshRefUrl("http://
thinkdigitserver.com/gettheme.php");
```

Modify the code for $formBlock in the renderPage() function in profile.inc to use <fb:ref> for the categories list and to update the list of available categories to all of our users at once:

```
  $formBlock = "
  <div class='sendFeedback'>
  <span class='sendTitle'>Send FeedBack</span>
  <form method='POST' id='FeedbackForm' action='".FB_APP_
URL."/'>
  <div class='categories'>
  <fb:ref handle='profileCategories'/>
  </div>
  <input type='hidden' name='target' value='$appUserID'/>
  <input class='textInput' name='Feedback'/><br/>
  <input      class='inputbutton'      type='submit'
name='submitFeedback'
  value='Send'>
  </form>
  </div>";
```

Let's now create a new updateCategoriesRef() function that will generate the FBML for the categories and stores it in a <fb:ref> handle.

```
  function updateCategoriesRef(){
  global $g_categories;
  $facebook = new Facebook(FB_API_KEY, FB_APP_SECRET);
  $categories = "";
  foreach( $g_categories as $name => $info ){
  $categories .="
  <div class='category'>
  <img class='categoryImg' title='{$info['title']}'
  src='".LOCAL_APP_URL."/img/".$info['smallimg']."'/><br>
  <div class='categoryTitle'>{$info['title']}</div>
  <input type='radio' name='category' value='$name' />
  </div>";
  }
  $categories .= "<div style='clear:both;'></div>";
  $facebook->api_client->fbml_setRefHandle("profileCatego
ries", $categories);
  }
```

Win!!

Exciting Prizes worth Rs.20000

What you learn from this Fast Track could win you Rs. 20,000 in exciting prizes. Take part in Digit's Facebook Apps contest by going to

www.thinkdigit.com/d/fb_contest

8 Feed stories

Facebook provides different ways for its applications to communicate and interact with its users and their friends. Basically, there are several ways to create and publish Feed stories, get them published to the stream and manage the story templates using both the Facebook tools and the API. There are three APIs to publish mini feeds and news feeds. But all of them are restricted to call not more than 10 times for a particular user in a 48 hour cycle. This means you can publish a maximum of 10 feeds in a specific user's profile in 48 hours:

feed_publishStoryToUser—The method used to publish a story to the news feed of any user (limited to call once every 12 hours).

feed_publishActionOfUser—The method used to publish the story to a user's mini feed, and to his or her friend's news feed (limited to call 10 times in a rolling 48 hour slot).

feed_publishTemplatizedAction—the method used to publish mini feeds and news feeds, but in an easier way (limited to call 10 times in a rolling 48 hour slot).

You can test this API from http://developers.facebook.com/tools.php?api by choosing Feed Preview Console.

8.1 Feed forms

Facebook Feed Form is the most basic method of publishing content on Facebook. We will start by creating a control that submits Feed stories to the stream. All you need to do to convert a normal HTML form into a Facebook Feed form is:

```
  <form    method='POST'    fbtype='multiFeedStory'
id='feedbackform'
  action='".LOCAL_APP_URL."feed_form_callback.php' >
  <h1>Select one of the readers and  your feedback.</h1>
  <input     class='inputbutton'     type='submit'
name='submitFeedback'
  label='Send Feedback' value='Send Feedback'/>
  </form>
```

The major change required here is that Feed forms should have an action attribute set to an absolute URL. In our case, it's been set to the full URL of a new file, feed_form_callback.php. Also we've added the fbtype attribute to the form tag. This makes Facebook to treat this form as a Feed form and publish its data to the stream.

As of now, there are two types of Feed forms: `feedStory` and `multiFeed-Story`. To create a feedStory form, you need to set the form's `fbtype` attribute to `feedStory` and to create `multiFeedStory` form, you need to set it to `multiFeedStory`. Although the differences between them are minor syntactic, the way they behave is completely different. The feedform publishes content to the current user's Wall and their friends' News Feeds, whereas the `multiFeedStory` form publishes it to both the Wall of the current user and one of his friends. For this reason, the multiFeedStory form also requires an FBML `<fb:friend-selector>` tag to be placed on the form to allow the user to choose to which of their friends' Walls they want to publish the content. Targeting multiple friends is not yet supported.

feed_publishTemplatizedAction is yet another API to publish news feed to a friend of the users of your application. It takes a number of parameters, and all the rules of publisActionOfUser are affected by the same kind of uncertainty and delay that is encountered while publishing the feed:

title_template - A template which contains the title of the feed

title_data - A JSON encoded string containing the value of the template variables

body_template - A template containing the text of the body

body_data - A JSON encoded string containing the value of the template variables

body_general - Additional template data to include in the body

image_1 - The URL of an image to be displayed in the Feed story

image_1_link - The URL destination after a click on the image referenced by image_1

image_2 - The URL of an image to be displayed in the Feed story

image_2_link - The URL destination after a click on the image referenced by image_2

image_3 - The URL of an image to be displayed in the Feed story

image_3_link - The URL destination after a click on the image referenced by image_3

image_4 - The URL of an image to be displayed in the Feed story

image_4_link - The URL destination after a click on the image referenced by image_4

target_id - Comma-delimited list of IDs of friends of the actor, used for stories about a direct action between the actor and these targets of his/her action. This is required if either the title_template or body_template includes the token `{target}`.

It can publish up to four images in the feed. If you want to make them clickable, supply the destination URL along with them. The following code snippet shows you how to use this API:

```
  <?php
  $titleTemplate + "{actor} gifts an {item} to {friend}";
  $titleData = array("item"=>"flower","friend"=>"<fb:name
uid='some uid'>");
  $bodyTemplate = "color of that {item} is {color}";
  $bodyData = array("color"=>"red","item"=>"flower");
  $image1 = "absolute url of an image";
  $image1Target = "target url when clicked over that
image";
  $facebook->api_client->feed_publishTemplatizedAction($
titleTemplate,json_encode($titleData),$bodyTemplate,json_
encode($bodyData),$image1,$image1Target);
  ?>
```

8.2 Publisher

Facebook offers several communication channels for applications and their users that can greatly expand an application's reach within the Facebook social graph, and ultimately, its potential for success, the basic being interacting with and producing content for the Facebook stream via Feed forms and the Facebook application programming interface (API). Another novel way for application users to publish content to the stream is via a special construct known as the Publisher which is more closely associated with an application user's profile and Home page than it is with an application. It allow your application users to insert new application content directly and immediately to the stream without having to leave their profile or Home page. Your Applications can register its own custom Publisher, and content created by it is always submitted to the stream in short story Feed format. It can even contain rich media, such as images, audio, or video.

There are several notification types, lightweight messages sent by applications without user interaction; which are one of the most powerful application communication mechanisms available. Finally, we delve into requests, which are one of the more controversial and misunderstood methods of application messaging. Requests are interesting, because they encompass one of the most direct means for applications to spread: invitations. You use all of your new, knowledge to update your sample application to use these features, which greatly enhances its capabilities.

The Publisher link of default Facebook applications always appear first in the list, followed by any application specific ones ordered by how recently they were used. If you visit a friend's profile, you see his Publishers arranged in an order that reflects the applications that your friend has used. If you click inside the Publisher box's text area, it expands it to show all the applications that have provided Publisher integration for the current profile.

Using Publisher, applications can present their own custom interactive interfaces for creating Feed content that are not subject to the same constraints that Facebook imposes on other application interaction points, such as profile boxes or application tabs. Your application can automatically play Flash videos or even hook the onload() event in Facebook JavaScript (FBJS).

Feed forms are specialised FBML forms that exist on application-specific canvas pages or application tabs. Facebook handles Feed forms in a way that client-side form validation is impossible. However, a Publisher interface has no such restrictions. A Publisher can use FBJS directly to dynamically enable a form's Submit button, depending on the state of the form's data or variables. When the application users employ the Publisher, they don't have to visit the application-specific page to publish application content. A Publisher can directly insert content into the Home page News feed and the Wall simultaneously and instantaneously. As soon as user submits a Feed story from a Publisher to his own profile or Home page, a short Feed story is created in both locations along with the Home page News feed of all of their friends.

8.3 Integrating with the publisher

Applications that provide publisher interfaces can provide two separate integration points:

● The first integration point used for publishing content to a user's own profile

● The second for publishing to friends' profiles.

Because Publishers are not required by the platform, applications can supply integration points for neither, one, or both. We'll update our application to provide both for illustrative purposes. We'll refer to the first flavour of Publisher as a Self-Publisher and the second, as an Other-Publisher.

8.4 Publisher developer settings

To get started with Publisher, you first have to go to your application's Developer Settings page (www.facebook.com/developers/apps.php), choose your application from the list on the left and then click on Edit Settings. Go to the Profiles tab and look for the section titled Profile Publisher. The options here are used to specify the text for the links used to activate the Publishers and the URLs from which Facebook will pull their content. For the Publish Text value, enter the name of the application, i.e., Compliment. For the Publish Callback URL, enter something like http://example.com/<application_name>/publisher_callback.php, but be sure to use your own local URL. If you decide to use different values for the link labels, make sure its less than 20 characters.

8.5 Creating the publisher

Once you have set the necessary options in your applications' Developer Settings, you can actually implement the Publisher. Facebook communicates with the application when using a Publisher by actually requesting information from the Publisher Callback URLs on two separate occasions:

- When a user clicks a Publisher link in a profile or Home page, Facebook makes calls to request the FBML and display the Publisher.
- If a user then submits a story from that Publisher, Facebook calls it again to request the content for the Feed story it has to publish. Your callback's code determines which type of request Facebook makes by checking the value of the method POST variable, which Facebook sends with each call.

The five POST variables sent by Facebook that you'll come across when creating Publishers are:

fb_sig_user - Facebook ID of user interacting with the Publisher. It is always sent.

fb_sig_profile_user - Facebook ID of user from whose profile the Publisher was requested. It is always sent.

fb_sig_session_key - If sent, it means the user identified by fb_sig_user has authorized the application that provided the Publisher.

app_params - Contains an array of the Publisher's form variable data, keyed by form element name. It is only sent when method has the value publisher_getFeedStory.

Now that you can determine if the values of fb_sig_user and fb_sig_profile_user are equal, it is certain that the user interacting with the Publisher is also the owner of the profile (or Home page) from which it was launched; we therefore need to display the Self-Publisher to allow this user to publish a Feed story to his own Wall and Home page News feed. If these values were to be different, you should display the Other-Publisher, which allows publishing to the Wall of the user identified by `fb_sig_profile_user` and his/her friends' Home pages. If you want the Publisher to offer enhanced functionality to users that have authorised the application that provides it, you can check for the existence of `fb_sig_session_key` in the POST variables which is only sent when this is true.

Create a new file named publisher_callback.php in your application root directory. The core behaviors of both displaying a custom Publisher and publishing a story to Facebook are shown in the following code block which you need to add to publisher_callback.php.

```
  <?php
  require_once 'inc/globals.inc';
  require_once 'inc/utils.inc';
  require_once 'inc/db.inc';
  require_once 'inc/profile.inc';
  require_once 'inc/notifications.inc';
  $fb = new Facebook(FB_API_KEY, FB_APP_SECRET);
  $publisher = $fb->user || $fb->fb_params['page_id'];
  $target = $fb->fb_params['profile_user']; //
  $isSelfPub = $publisher === $target;
  $publisherAction = $_POST['method'];
  $fbResponse = '';
  if( isset( $publisherAction ) &&
  isset( $publisher ) &&
  isset( $target ) ) {
  if( 'publisher_getInterface' === $publisherAction ) {
  $markup = getPublisherUI($isSelfPub ? $publisher :
$target);
  $fbResponse = array( 'content'=> array( 'fbml' =>
$markup, 'publishEnabled' => true ), 'method' =>
$publisherAction );
  } else if ( 'publisher_getFeedStory' === $publisherAction
) {
  $formValues = $_POST['app_params'];
  if( isset( $formValues ) ) {
  global $g_categories;
  if(!isset($fb->fb_params['page_id'])) {
```

```
  $db = new DB();
  $db->addCompliment( $publisher, $isSelfPub ? $publisher :
$target,$formValues['category'],$formValues['compliment']
);
  }
  $categoryInfo = $g_categories[$formValues['category']];
  $imageSrc          =          LOCAL_APP_URL.'/
img/'.$categoryInfo['bigimg'];
  $imageLink = LOCAL_APP_URL;
  $images = array('src'=> $imageSrc, 'href'=> $imageLink);
  $feed = array( 'template_id' =>
  $isSelfPub ? TEMPLATE_BUNDLE_SELF_PUBLISH_1 : TEMPLATE_
BUNDLE_OTHER_PUBLISH_1, 'template_data' => array( 'app' =>
'<a href="'.FB_APP_URL. '">Compliments</a>',
  'ctitle' => $categoryInfo['title'],
  'ctext' => $formValues['compliment'],
  'images' => array( $images ) ) );
  } else {
  $fbResponse  =  getPublisherError('Incorrect  method
requested:  Expected  either  publisher_getFeedStory  or
publisher_getInterface');
  }
  $fbResponse = array( 'method' => $publisherAction,
  'content' => array( 'feed' => $feed ) );
  } else {
  $fbResponse  =  getPublisherError('Missing  form  values
from Publisher');
  }
  } else {
  $fbResponse = getPublisherError('Either method, user, or
target are not specified');
  }
  echo json_encode($fbResponse);
  ?>
```

You are calling the `getPublisherError()` function to create arrays that Facebook uses to notify your users of specific error conditions. You must also provide Facebook with a JSON-encoded associative array containing three keys: errorCode (which must always be set to 1, handily defined by the constant FACEBOOK_API_VALIDATION_ERROR), errorTitle, and errorMessage. The values for errorTitle and errorMessage are used to populate the error dialog's title and body content, respectively. Add the following code block to the end of publisher_callback.php to achieve that:

```
  function getPublisherError($msg) {
  return array( 'errorCode' => FACEBOOK_API_VALIDATION_
ERROR,
  'errorTitle' => 'Facebook Publisher Error',
  'errorMessage'=> $msg );
  }
```

You are using the getPublisherUI() function to provide the markup for the Publisher's user interface, which Facebook requests after sending the publisher_getInterface POST variable. getPublisherUI() uses the getCSS() function to provide the styles for the Publisher. Add the following code block to the end of publisher_callback.php.

```
  function getCSS() {
  $css = "
  <style>
  h1 { font-size: 14px; font-weight:bold; text-align:center;
margin-top:6px;
  margin-bottom:6px; }
  .panel { text-align:center; background-color:#F7F7F7;
padding:10px 0;}
  #complimentTable { margin: 5px auto; }
  #complimentTable .textInput { width: 85%; }
  #complimentTable td { text-align:center; }
  .category { float:left; height:75px; text-align:center;
margin:8px; }
  .category img { width:48px; height:48px; }
  .category .categoryTitle { height:20px; font-weight:bold;
}
  </style>";
  return $css;
  }
  function getPublisherUI( $uid ) {
  global $g_categories;
  $out = getCSS();
  $out .= "
  <div class='panel'>
  <form>
  <table id='complimentTable'>
  <tr>
  <td>
  <fb:if-is-friends-with-viewer       uid='$uid'
includeself='false'>
  <h1>
  <fb:name uid='$uid' firstnameonly='true' linked='false'/>
is:
  </h1>
  <fb:else>
  <h1>
  <fb:name  uid='$uid'  capitalize='true'  linked='false'/>
are:
  </h1>
  </fb:else>
  </fb:if-is-friends-with-viewer>
  </td>
  </tr>
  <tr>
  <td>";
  foreach( $g_categories as $name => $info ){
  $out .= "
  <div class='category'>
  <img class='catImg'
  src='".LOCAL_APP_URL."/img/{$info['bigimg']}'/><br>
  <span     class='categoryTitle'>{$info['title']}</
span><br/>
  <input type='radio' name='category' value='$name'/>
  </div>";
  }
  $out .= "
  <div style='clear:both;'></div>
  </td>
  </tr>
  <tr>
  <td>
  <fb:if-is-friends-with-viewer        uid='$uid'
includeself='false'>
  <h1>because they:</h1>
  <fb:else>
  <h1>because you:</h1>
  </fb:else>
  </fb:if-is-friends-with-viewer>
  </td>
  </tr>
  <tr>
  <td>
  <input class='textInput' name='compliment' />
  </td>
  </tr>
  </table>
  </form>
  </div>";
  return $out;
  }
```

There are sections in this code that deal with responding to the receipt of the method variable in the POST variables received from Facebook. If the value of method is publisher_getInterface, your code will provide Facebook with a specific JSON response that provides it with the raw FBML that comprises the Publisher's user interface, as follows:

```
  if( 'publisher_getInterface' === $publisherAction ) {
  $markup  =  getPublisherUI($isSelfPub  ?  $publisher  :
$target);
  $fbResponse  =  array(  'content'=>  array(  'fbml'  =>
$markup,
  'publishEnabled' => true ),
  'method' => $publisherAction );
  }
```

9 Invitations and notifications

Facebook allows your application user to invite his friends to use that application, which acts like viral marketing in spreading your application. So, as an application developer, you should determine when and how to let them invite their friends and how to make your application a sure success so that the friends of your application user also pay a visit to your application. You can also let their user send email notifications. Let's get started on creating a successful invitation and notification system for your application, something which uses these constructs efficiently and help your application spread.

9.1 Invitations

Invitations are used to send emails to friends of a user of that application in Facebook applications, using which, any one of his or her friends can subscribe to your Facebook application. This is a great way of promoting your application.

Until sometime back, users of your application could to send invitations to all their friends at a time, but recently Facebook added some strict policies such as, a user can invite only 20 of his/her friends each day and so you have to design the whole system using Facebook API, or the default widget provided. An invitation system on Facebook has several parts. The FBML tags used to create an Invitation system are:

- fb:request-form
- fb:req-choice
- fb:request-form-submit
- fb:application-name
- fb:multi-friend-input
- fb:friend-selector
- fb:multi-friend-selector(condensed)

However, you cannot change the default look and feel of this one unless you use CSS style sheets.

9.1.1 fb:request-form

This tag acts as the primary widget for inviting friends A typical invitation code will look like:

```
<fb:request-form
```

```
  action="invitation processor page"
  method="POST"
  invite="true"
  type="Application Name"
  content="Content of your Mail">
  <fb:multi-friend-selector
  showborder="false"
  actiontext="Invite friends to the Application Name"
  exclude_ids="exclude friends who are already using
this app"
  max="20" />
  </fb:request-form>
```

The various parameters for this FBML tag are:

- **Action** - This parameter supplies the information as to where the page will submit all the information when someone clicks on the send invitation or skip button.
- **Method** - This parameter is used to select GET or POST, the way information will be submitted.
- **Invite** - It can be true or false. When set as true, the invitation will go as invitation, otherwise it will go as a request. This change actually reflects the content of the invitation that the user will get.
- **Type** - This one is used to reflect the context of your application. It is usually set the same as the application name.

9.1.2 fb:req-choice

This tag is used to insert a button in your request message, and define the name and target URL. So when a user receives your invitation, they will also see this button, and when they click on it, they will be redirected to the URL that you have specified. This is very useful if you want them to install the application, just by providing a nice button inside your invitation message.

The various parameters for this FBML tag are:

- url - This is the target URL which used to redirect the user on a click event.
- Label – This is just the caption for the button.

Example:

```
  <fb:req-choice url="http://apps.facebook.com/example/
Accept.php" label="Accept" />
```

Don't use "ignore" as the caption of any request button because it is there by default. So if you insert another ignore button, the code will not work and you might lose all the other buttons inside too.

9.1.3 fb:request-form-submit

This tag is used to display a submit button inside your invitation form.The various parameters for this FBML tag which let you display the user's name or the full name inside this button are:

- uid - The user id of the user you want to send this invitation to.
- label – Used to provide a caption to the button. Tip -To show the first name of the user, use %n and to show the full name, use `%N`.

Example:

```
<fb:request-form-submituid= 'some uid' label= 'send
%N an Invitation' />
```

9.1.4 fb:multi-friend-selector

This FBML tag is used to draw a friend selector widget anywhere in your canvas page. The various parameters for this FBML tag are:

- **actiontext** –Used to specify what to display at the top of this friend selector as the heading.
- **showborder** – Can be set to true or false.
- **rows** – Used to specify the number of rows, that friends will be shown in. The default value is 5. But you can choose anything from 3 to 10.
- **max** – Used to specify the maximum number of friends a user can select at a time to send an invitation to. It is advisable to keep it at 20 because of Facebook regulations.
- **exclude_ids** – This is a comma separated list of friends who are not to be shown inside this selector object. Use this to exclude all your friends who have already installed this application.
- **bypass** – Used to add a label to the skip button.
- **condensed** – If set to true, it will create a condensed page that looks far different from a full version.

9.1.5 fb:multi-friend-input

Instead of using a multi-friend selector in your Invitation system, you can also use a multi-friend input in a fb:request-form, although it is much less user-friendly as compared to a friend selector. The usage of multi-friend selector is very simple and you can select an arbitrary numbers of friends.

The various parameters for this FBML tag are:

- **width** – Used to specify the width of the input box.
- **border_color** – Used to enter the HTML colour code to colour the border of this selector panel.
- **include_me** - If this is set to true, the user can select himself/herself. By default, it is obviously set to false.
- **max** – Used to specify the maximum number of friends a user can select at a time to send an invitation to. It is advisable to keep it at 20 because of Facebook regulations.
- **exclude_ids** - This is a comma separated list of friends who are not to be shown inside this selector object. Use this to exclude all your friends who have already installed this application and they will be excluded from the auto drop-down box.
- **prefill_ids** - This is a comma separated list of friend ids, who will be selected automatically whenever this box is viewed.
- **prefill_locked** - If this is set to true, the user cannot remove the prefilled friends.

Example:

```
<fb:multi-friend-input width="300px" border_
color="#000" exclude_ids="some id, another id" />
```

You might want to manage the data that the user inputs. This can be possible because if your form contains multi-friend input; once it is submitted, you will find all the IDs of the selected friends in $_POST['ids'] or $_GET['ids']

9.1.6 fb:application-name

This FBML tag is used to easily define the name of your application, especially when it contains some restricted words in it, such as "message". Facebook limits the usage of these restricted words when invitations go to news feed and mini feed, so you need to use fb:application-name to rescue them.

Example:

```
<fb:application-name>
Message builder
</fb:application-name>
```

9.2 Notifications

There are two ways of sending notifications to your users: By using the Notify page or by an email. Sending notifications to the notification page of a user does not require confirmation from the specific user. So you can use this to send information about the activity of their friends who are also users of your application to them. You can also send notification to non-users of your applications, but it can be classified as spam. There are two functions in the Facebook REST API useful for sending notifications:

- notifications_send
- notifications_sendEmail

9.2.1 notifications_send

This function is used to send notifications to the user's notifications page, located at http://www.facebook.com/notifications.php. The following source code sends invitations to your application.

```
<?php
include_once("prepend.php");
$facebook->api_client->notifications_send(array(503274632),"This is a sampleNotification from Digit Sample App");
?>
```

This function has two parameters : an array of users to whom you want to send notifications and a text block, which can contain only text and links.

9.2.2 notifications_sendEmail

This function is used to send notifications to a user's mailbox, instead of the notification page. But remember that Facebook allows you to send only a maximum number of 5 emails per user, per day. This also needs the user's permission before you finally dispatch the mail. Let's take a look:

```
<div style="padding:20px;">
<?php
if (empty($_POST['ids'])){
?>
<form method="POST">
Select Some friends whom you want to send an Email.
<fb:multi-friend-input width="300px" border_color="#000" include_me='true' exclude_ids="" /><br/>
Type your message to them<br/>
<textarea name ='email'></textarea><br/>
<input type='submit' value='Send Mail' />
</form>
```

```
  <?}?>
  </div>
  <div style="padding:20px;">
  <?php
  include_once("prepend.php");
  if (!empty($_POST['ids'])){
  $email = $_POST['email'];
  $successful_friends=     $facebook->api_client-
>notifications_sendEmail($_POST['ids'],"This  is  a
sample  Mail  from  Facebook  Application  Development
Book",$email,$email);
  echo  "Emails  successfully  sent  to  the  following
friends <br/>";
  $parts = explode(",",$successful_friends);
  foreach($parts as $part)
  echo "<fb:nameuid='{$part}' useyou='false' /></br>";
  }
  ?>
  </div>
```

This function has four parameters:

- An array of users whom you want to email.
- The subject of the email.
- The content in plain text.
- The content in HTML.

This function also returns a comma separated list of all successful recipients. As notification and emails are the most effective way of keeping your users up to date as well as promoting your application, you should use them carefully to take maximum advantage of them. Make sure you catch the most interesting moments of your application, and ask your users to send emails to their friends before it. For example, if you have created a quiz application; after displaying the result to your user who has taken the quiz just now, ask them to compare with their friends and display an invite page. Most of the users will invite their friends in such a case. In the email you are sending, you can send the score of that user and challenge him to compare which makes his invitation more enticing. Also as soon as someone installs your application, show them an invitation page before they dive further. Keep random links such as "Show your performance/score/Compare whatever to your friends", which redirect the users to an invitation page when clicked. Whenever a user has achieved or done something interesting, send his friends some notification regarding it. It generally works as viral marketing. You can keep a page in your application, where a user can monitor which of his friends also use it and track the activities of his friends for that application. It all depends on your creativity. d

digit

thinkdigit.com

Digit's
Facebook Apps
Contest

What you learn from this Fast Track could win you Rs. 20,000 in exciting prizes. Take part in Digit's Facebook Apps contest by going to

www.thinkdigit.com/d/fb_contest

10 Additional Resources

Now, that you've started and built your application, there will be certain problems you will run into while implanting the various features we talked about. So,what do you do? The single best resource for these types of problems is the Facebook Developers forum at http://forum.developers.facebook.com/. There's also a channel on freenode (#Facebook) if you're old-school and more comfortable on IRC. This is at times the faster way to get an answer to a specific question. You can also check out the different Facebook developer groups (http://wiki.developers.facebook.com/index.php/Local_Developer_Group).

Besides these are are several good references for Facebook Developers:

- All Facebook. http://allfacebook.com
- Developer News. http://developers.facebook.com/news.php
- Facebook News. http://blog.facebook.com/blog.php
- Inside Facebook. www.insidefacebook.com
- Application Verification Program. http://developers.facebook.com/verification.php
- Client Libraries. http://wiki.developers.facebook.com/index.php/Client_Libraries
- Custom Tags Directory. http://wiki.developers.facebook.com/index.php/Custom_Tags_Directory
- Developer Application. www.facebook.com/developers
- Developer Wiki. http://wiki.developers.facebook.com/index.php/Main_Page
- Developer Forum. http://forum.developers.facebook.com/
- Developer Test Accounts. www.facebook.com/developers/become_test_account.php
- Developer Test Consoles. http://developers.facebook.com/tools.php
- Facebook Open Source. http://developers.facebook.com/opensource.php
- fbFund. http://developers.facebook.com/fbFund.php?tab=about
- Guiding Principles. http://developers.facebook.com/get_started.php?tab=principles
- Platform Statistics. www.facebook.com/press/info.php?statistics
- Terms of Service. www.facebook.com/terms.php
- APC. http://php.net/apc
- Firebug. www.getfirebug.com
- Firebug Lite. http://getfirebug.com/lite.html
- Memcached. www.danga.com/memcached
- Web Developer Toolbar. http://chrispederick.com/work/web-developer
- YSlow. http://developer.yahoo.com/yslow
- YUI Library. http://developer.yahoo.com/yui

10.1 Taking your application further

Facebook platform also supports various resources that help you analyze your application statistics, where to go if you get stuck, and, perhaps most important, how to generate a revenue stream from your application! A lot of applications generate a good revenue stream from advertisements and with some planning and some good choices, you should be able to at least offset the costs of your server hosting, if not turn into a millionaire.

10.1.1 Application Statistics

There is a basic statistics feature in the Facebook Developer application to help you get an idea of what's going on with your application by giving you detailed stats on usage, HTTP status requests, and recent HTTP requests. You are also provided with a helpful User Engagement statistic that tells you how many people have used your application in the past 24 hours. This helps you figure out how many people are actually "using" your application.

But, if you need or want more sophisticated statistics, the `<fb:googleanalytics>` tag comes in handy. Create an account on Google Analytics to use this tag and it will provide you with exceptionally detailed statistics about your application (and for that matter, any web site build). Navigate to http://www.google.com/analytics/ and sign in with your Google account to use Google Analytics. Add a web site profile by clicking on the Add Website Profile link to start the process. Next, fill out the form for a new domain using your server address URL (not your http://apps.facebook.com/<your_app_name> URL) and click on continue. You will now be presented with a small snippet of JavaScript to add to your page. But it's not FBJS. So, you cannot add it directly. Just get the account number (defined by the _uacct variable) from that code to use the `<fb:google-analytics>` tag, and let Facebook write this in for you.

Add the following code in your application to produce the required JavaScript in your application:

```
  <fb:google-analytics  uacct="<your_UA_account_number>"
/>
```

Facebook will automatically correct and add the JavaScript to the resultant HTML stream. You will also have access to the Google Analytics tracker in FBJS with the Facebook.urchinTracker object in case you want it. <fb:google-analytics> tag is the easiest way to implement your own methods through FBJS; however, if you should you need more granular control over what gets sent to the Google servers for your application, the functionality does exist.

Google Analytics provides you with a really great statistics set including a site overlay, geotargeting, Google AdWords integration, and just about any other type of useful statistic that you could possibly want about the people using your application.

10.1.2 Monetizing your application

Facebook has several service agreements which allow application developers to monetize their applications. The most popular among them is Google Adsense.

AdSense

You've already set up a Google Analytics account in the previous section. Now to enable Google Adsense too, all you need to do is - go to the AdSense web site at http://adsense.google.com, and create a separate AdSense account by clicking on the Sign Up Now button (if you don't already have an account). Once you've set up an AdSense account and filled out the appropriate forms, you get the option to decide what type of advertising you want to implement from among ads for content, search, referrals, video, and mobile content. Google also provides you with some other options for your ads (text and images or text or images only), size and color palette etc. so choose carefully a theme that matches your overall application design. The best aspect AdSense is that its ads are generally unobtrusive and can be placed anywhere on a page. There are also nice tools for tracking your earnings and growth over time.

To get relevant ads from Google, you first need to make your canvas page publicly available so it can be crawled by Google. To do this, just change all the require_login functions in the application to the get_loggedin_user function:

```
$user = $facebook->get_loggedin_user();
```

You can use any of the previously used canvas page tags on this page without any changes to your code, although you should refactor some of your existing code for non-logged-in users.

There is also an iframe way to use AdSense. Make a new page that contains the JavaScript that Google provides you when you generate the ads for your page on your web site, and then call that page through the <fb:iframe> tag on your canvas page.

```
<fb:iframe src="http://siteURL.com/ads.php"
width="<google_ad_width>"
  height="<google_ad_height>" />
```

This is the code snipped you need to insert in your code where you want to place the ads for AdSense to work properly on iFrame pages.

Some others ways to generate revenue

There are a lot of other services to generate revenue from your Facebook application. Most of these work just like AdSense and leverage their advertising in your application. Here is a list of some of them that you might want to check out:

- AdBrite (http://www.adbrite.com)
- Appsaholic (http://apps.facebook.com/appsaholic/)
- BannerConnect (http://www.bannerconnect.net)
- Chitika (http://chitika.com/facebookapi.php)
- Cubics (http://www.cubics.com)
- fbExchange (http://fbexchange.com)
- Neverblue (http://www.neverbluemedia.com)
- PeanutLabs (http://www.peanutlabs.com)
- Survey Savvy (http://www.surveysavvy.com)
- Zohark Ads (http://www.facebook.com/applications/Zohark_Ads/18584639088)

If I've missed your company, sincere apologies!

10.1.3 Selling Your Application

Another great way to make some money from your application is to sell it. Build your application to maturity and show some sustained growth or come up with some type of novel method that has a lot of promise to attract buyers. If you think you've developed a killer application that has a lot of potential and, for whatever reason, you want to sell your stake, there are a few websites to start looking for a buyer:

- Altura (http://altura.com/)
- AppFactory (http://www.baypartners.com/appfactory/)
- EBay (http://www.ebay.com/) d